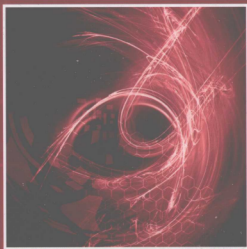




普通高等教育“十一五”计算机类规划教材

Visual Basic 程序设计

王怀彬 等编著



免费
电子课件



机械工业出版社
CHINA MACHINE PRESS



普通高等教育“十一五”计算机类规划教材

20882	协同软件技术及应用	汤庸 冀高峰	朱 庸
20998	形式语言与自动机理论 (免费电子课件)	吴哲辉	吴振寰
22780	数据库原理及应用 (免费电子课件)		胡孔法
21424	计算机专业英语 (免费电子课件)		霍宏涛
21547	微机原理与接口技术 (免费电子课件)		吉海彦
22750	IBM—PC 汇编语言程序设计 (免费电子课件)		余朝琨
22779	C#.NET 程序设计 (免费电子课件)		李旗
22781	ATmega 系列单片机原理及应用——C 语言教程 (免费电子课件)		海 涛
23147	Linux 系统应用基础教程 (免费电子课件)		张小进
23185	计算机网络技术与应用 (免费电子课件)		刘冰
23200	多媒体技术基础及应用 (免费电子课件)		刘建
23201	数据结构 (免费电子课件)		戴敏
10423	计算机文化基础教程 (第 2 版) (免费电子课件)		吕昌玲
23298	电路与电子技术基础 (免费电子课件)	王金矿 李心广	张晶
23502	Visual Basic 程序设计 (免费电子课件)		王怀彬
23535	离散数学及其应用 (免费电子课件)		魏雪丽
[]	算法设计方法		吴哲辉
[]	C++ 程序设计		张桦
[]	网络编程与计算技术		刘化君
[]	数字化摄影技术		穆强
[]	C++ 程序设计与应用		周仲宁
[]	C++ 程序设计与应用实验指导及习题解析		周仲宁
[]	Access 数据库基础及应用教程		米红娟
[]	网页设计与制作		王任华
[]	汇编语言与计算机系统组成		李心广
[]	计算机软件技术实训教程		冯元椿

编辑热线: (010)88379726

● ISBN 978-7-111-23502-6
● 策划：刘丽敏 / 封面设计：张静

定价：35.00 元

ISBN 978-7-111-23502-6



9 787111 235026 >

普通高等教育“十一五”计算机类规划教材

Visual Basic 程序设计

王怀彬 杨文军 冯东晖 韩盛磊 卜宏津编著

2. 创建安装文件

设置(910)自定义安装图

的步骤如下:

- 1) 为 VB 添加“打包和安装向导”。在“VB 工程”窗口中，依次打开菜单“外
“外程序管理器”窗口，单击“打包和安装向导”按钮，单击“确定”



机械工业出版社

图 13-14 “外程序管理器”窗口

本书是普通高等教育“十一五”计算机类规划教材。全书共分13章,以 Visual Basic 6.0 版为主,从零开始逐步介绍了 Visual Basic 程序设计的整个过程。主要内容包括: Visual Basic 的功能和特点、开发环境、联机帮助文档(MSDN)、面向对象的概念及创建一个 Visual Basic 应用程序的步骤; Visual Basic 语言的编程基础,包括数据类型、常量、变量、运算符及表达式、常用的内部函数及注释和代码的书写规范;输入/输出操作、窗体、常用的内部控件、ActiveX 控件及第三方控件的开发和使用; Visual Basic 程序设计中的3种程序结构,包括顺序、分支和循环结构;数组、数组的基本操作、多维数组及控件数组; Sub 过程、Function 过程、参数传递、过程调用及嵌套调用;绘图程序的设计方法、绘图中的基本控件及鼠标和键盘操作;文件、顺序文件和随机文件的操作、文件系统控件;网络、网络相关的控件和多媒体控件;数据库的基础知识、SQL 语句、DAO 和 ADO 控件、DAO 对象和 ADO 对象及其应用;菜单、工具栏和状态栏及其应用;MDI 窗体及其应用; Visual Basic 程序的调试方式及创建安装文件方式等。

本书可作为高等院校计算机科学与技术相关专业的教材,也可作为广大计算机爱好者和工程技术人员的参考书。为方便教师教学,本书配有教学课件,欢迎选用本书作为教材的老师来信索取,索取邮箱: llm7785@sina.com。

图书在版编目(CIP)数据

Visual Basic 程序设计/王怀彬等编著. —北京:机械工业出版社, 2008.4

普通高等教育“十一五”计算机类规划教材
ISBN 978-7-111-23502-6

I. V… II. 王… III. BASIC 语言—程序设计—高等学校—教材
IV. TP312

中国版本图书馆 CIP 数据核字(2008)第 020238 号

机械工业出版社(北京市百万庄大街 22 号 邮政编码 100037)

责任编辑:刘丽敏 责任校对:姜 婷

封面设计:张 静 责任印制:洪汉军

北京铭成印刷有限公司印刷

2008 年 4 月第 1 版第 1 次印刷

184mm × 260mm · 21.75 印张 · 538 千字

标准书号: ISBN 978-7-111-23502-6

定价: 35.00 元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

销售服务热线电话: (010)68326294

购书热线电话: (010)88379639 88379641 88379643

编辑热线电话: (010)88379726

封面无防伪标均为盗版

前 言

Visual Basic 是 Microsoft 公司开发的一个功能强大的编程工具,它继承了 BASIC 语言简单易学的特点,又增加了许多新的功能。它采用面向对象与事件驱动的程序设计思想,使编程更加方便、快捷,即使没有接触过程序开发的人员,也能够较短时间内利用它开发出功能完善的应用程序。使用 Visual Basic 既可以开发多媒体应用程序、网络应用程序,同时也可以开发基于数据库应用的应用程序。

本书通过大量的实例,以 Visual Basic 6.0 版为主,从零开始逐步介绍了 Visual Basic 程序设计的整个过程。在编写风格上尽量避免枯燥的理论讲解,使读者容易上手,能够快速地掌握程序设计的思想和方法,做到有的放矢。

针对初学者的特点,本书在排版上注意由易到难、由浅入深和循序渐进的方式,力求通俗易懂、简洁实用。本书概念清晰、逻辑性强,层次分明、例题丰富,符合教师教学和学生学习的习惯。

全书共分 13 章,各章的内容如下:

第 1 章主要介绍了 Visual Basic 的功能和特点、开发环境、联机帮助文档(MSDN)、面向对象的概念及创建一个 Visual Basic 应用程序的步骤。

第 2 章主要介绍了 Visual Basic 语言的编程基础,包括数据类型、常量、变量、运算符及表达式、常用的内部函数及注释和代码的书写规范。

第 3 章主要介绍了输入/输出操作、窗体、常用的内部控件、ActiveX 控件及第三方控件的开发和使用。

第 4 章主要介绍了 Visual Basic 程序设计中的 3 种程序结构,包括顺序、分支和循环结构。

第 5 章主要介绍了数组、数组的基本操作、多维数组及控件数组。

第 6 章主要介绍了 Sub 过程、Function 过程、参数传递、过程调用及嵌套调用。

第 7 章主要介绍了绘图程序的设计方法、绘图中的基本控件及鼠标和键盘操作。

第 8 章主要介绍了文件、顺序文件和随机文件的操作、文件系统控件。

第 9 章主要介绍了网络、网络相关的控件和多媒体控件。

第 10 章主要介绍了数据库的基础知识、SQL 语句、DAO 和 ADO 控件、DAO 对象和 ADO 对象及其应用。

第 11 章主要介绍了菜单、工具栏和状态栏及其应用。

第 12 章主要介绍了 MDI 窗体及其应用。

第 13 章主要介绍了 Visual Basic 程序的调试方式及创建安装文件方式。

本书可作为高等院校计算机科学与技术及相关专业学生的教材,也可作为广大计算机爱好者和工程技术人员参考书。

本书由王怀彬、杨文军、冯东晖、韩盛磊、卜宏津编著。王怀彬编写了第 5 章,杨文军编写了第 6、10、11、13 章,冯东晖编写了第 1、3、4 章,韩盛磊编写了第 7、8、9 章,卜

由于编者水平有限,难免有错误之处,希望广大读者批评指正,多提宝贵意见。

編者

目 录

前言

第1章 Visual Basic 概述 1

1.1 Visual Basic 6.0 简介 1

1.2 Visual Basic 主要功能和特点 1

1.3 Visual Basic 可视化编程环境 2

1.3.1 窗体窗口 3

1.3.2 窗体布局窗口 4

1.3.3 工程资源管理器窗口 4

1.3.4 工具箱 4

1.3.5 属性窗口 5

1.3.6 菜单 5

1.3.7 工具栏 6

1.4 Visual Basic 联机帮助 6

1.4.1 MSDN 的安装 6

1.4.2 MSDN Library 的使用 7

1.4.3 上下文相关帮助 7

1.4.4 运行帮助系统实例 8

1.5 面向对象的基本概念 8

1.5.1 什么是面向对象 9

1.5.2 对象的属性 9

1.5.3 对象的事件 10

1.5.4 对象的方法 10

1.6 一个简单的 Visual Basic 程序 11

1.6.1 创建应用程序界面 11

1.6.2 设置对象的属性 12

1.6.3 编写代码 13

1.6.4 保存文件和打开文件 13

1.6.5 在 Visual Basic 环境中运行程序 14

1.6.6 生成可执行文件 15

习题 15

第2章 Visual Basic 语言基础 17

2.1 数据类型 17

2.1.1 基本数据类型 17

2.1.2 用户定义数据类型 18

2.2 常量 19

2.2.1 直接常量 19

2.2.2 系统内部常量 21

2.2.3 符号常量 22

2.2.4 常量的命名 23

2.3 变量 23

2.3.1 变量的声明 23

2.3.2 变量的存取 25

2.3.3 变量的作用域 25

2.3.4 变量名 26

2.3.5 变量的数据类型转换 27

2.4 运算符和表达式 28

2.4.1 运算符 28

2.4.2 表达式 34

2.4.3 运算符的优先级 35

2.5 常用的内部函数 36

2.5.1 输入函数 InputBox 36

2.5.2 输出函数 MsgBox 38

2.5.3 类型转换函数 42

2.5.4 数学函数 42

2.5.5 日期与时间函数 44

2.5.6 字符串函数 46

2.5.7 目录和文件函数 51

2.5.8 数组函数 52

2.5.9 其他常用函数 53

2.6 Visual Basic 程序的注释和书写规范 54

2.6.1 Visual Basic 程序的注释 54

2.6.2 Visual Basic 程序的书写规范 55

习题 56

第3章 窗体和控件 58

3.1 赋值语句和赋值相容 58

3.1.1 赋值语句 58

3.1.2 赋值相容 59

3.1.3 Set 语句 59



3.2 输入/输出操作	60	4.3 循环结构程序	109
3.2.1 输入操作	60	4.3.1 For 循环	109
3.2.2 输出操作	60	4.3.2 While 循环	111
3.2.3 常用的输出格式控制函数 和方法	62	4.3.3 Do 循环	112
3.3 窗体	65	4.3.4 循环嵌套	115
3.3.1 主要属性	65	4.3.5 中途跳出循环	116
3.3.2 主要事件	65	4.3.6 循环语句的小结	118
3.3.3 主要方法	66	习题	121
3.3.4 主要语句	67	第5章 数组	122
3.3.5 窗体的生命周期	67	5.1 数组概述	122
3.4 常用的内部控件	68	5.2 数组声明	123
3.4.1 命令按钮	69	5.2.1 固定大小的数组的定义	123
3.4.2 标签	70	5.2.2 动态数组的定义	124
3.4.3 文本框	71	5.3 数组操作	125
3.4.4 框架	73	5.3.1 数组元素的引用	126
3.4.5 复选框	74	5.3.2 给数组元素赋初值	126
3.4.6 单选按钮	75	5.3.3 数组的输入	127
3.4.7 列表框	76	5.3.4 数组的输出	127
3.4.8 组合框	80	5.3.5 数组元素的复制	128
3.4.9 滚动条	82	5.3.6 数组的清除	129
3.4.10 计时器	84	5.3.7 针对数组的循环语句 For Each...Next	130
3.5 ActiveX 控件	85	5.4 多维数组	131
3.5.1 通用对话框	86	5.4.1 多维数组的声明	131
3.5.2 进度条控件	90	5.4.2 多维数组的操作	131
3.5.3 图像列表控件	91	5.4.3 保留动态数组的内容	133
3.5.4 选项卡控件	92	5.4.4 数组边界的检测函数	134
3.6 第三方控件的使用	95	5.5 控件数组	134
习题	96	5.5.1 控件数组的概念	135
第4章 程序控制结构	97	5.5.2 控件数组的创建方法	135
4.1 顺序结构程序	97	5.6 数组的应用	137
4.1.1 End 语句	97	习题	153
4.1.2 Rem 语句	97	第6章 过程	155
4.2 分支结构程序	98	6.1 Sub 过程	155
4.2.1 If 语句	98	6.1.1 Sub 过程的定义	155
4.2.2 Select Case 语句	103	6.1.2 Sub 过程的创建	156
4.2.3 分支嵌套	105	6.1.3 Sub 过程的调用	157
4.2.4 GoTo 语句	106	6.2 Function 过程	158
4.2.5 支持分支选择的函数和控件	107	6.2.1 Function 过程的定义	158



6.2.2 Function 过程的创建	159	8.4.2 目录列表框	235
6.2.3 Function 过程的调用	160	8.4.3 文件列表框	235
6.3 过程的参数传递	161	8.5 文件操作应用	237
6.3.1 形式参数和实际参数	161	8.5.1 传统的文件处理机制	237
6.3.2 按值传递和按地址传递参数	162	8.5.2 文件系统对象处理机制	240
6.3.3 传递数组	163	习题	252
6.4 过程的嵌套与递归调用	164	第 9 章 网络和多媒体	253
6.4.1 过程的嵌套调用	164	9.1 网络概述	253
6.4.2 过程的递归调用	165	9.2 网络控件	255
6.5 应用举例	165	9.2.1 Winsock 控件的属性、方法和事件	255
6.5.1 查找问题	165	9.2.2 Winsock 控件的使用	262
6.5.2 插入问题	167	9.3 多媒体控件	269
习题	168	9.3.1 Animation 控件	269
第 7 章 绘图	170	9.3.2 Multimedia MCI 控件	274
7.1 绘图的相关知识	170	9.4 应用实例	279
7.1.1 坐标系统	170	习题	285
7.1.2 与绘图相关的属性	175	第 10 章 数据库	286
7.1.3 颜色	178	10.1 数据库基础知识	286
7.2 绘图操作	181	10.1.1 数据库基本概念	286
7.2.1 图形控件	181	10.1.2 关系数据库	287
7.2.2 绘图方法	183	10.2 SQL 语言	288
7.3 键盘和鼠标操作	188	10.3 数据控件及应用	290
7.3.1 键盘事件	189	10.3.1 Data 控件	290
7.3.2 鼠标事件	191	10.3.2 ADO 控件	291
7.4 应用举例	195	10.3.3 数据控件的记录集 RecordSet 对象	293
习题	205	10.3.4 数据绑定控件	295
第 8 章 文件操作	207	10.3.5 数据控件应用	295
8.1 文件	207	10.4 数据访问对象及应用	298
8.1.1 文件结构	207	10.4.1 DAO 数据访问对象	298
8.1.2 文件类型	208	10.4.2 ADO 数据访问对象	300
8.1.3 文件的打开与关闭	208	10.4.3 数据访问对象的应用	301
8.2 顺序文件	211	习题	309
8.2.1 顺序文件的写操作	211	第 11 章 菜单、工具栏与状态栏	310
8.2.2 顺序文件的读操作	215	11.1 菜单	310
8.3 随机文件	221	11.1.1 菜单的构成	310
8.3.1 随机文件的读/写操作	222	11.1.2 下拉菜单	311
8.3.2 记录操作	228	11.1.3 上下文菜单	314
8.4 文件系统控件	234		
8.4.1 驱动器列表框	234		



11.2	工具栏	316
11.3	状态栏	318
11.4	综合应用实例	320

习题	323
----	-----

第 12 章 MDI 窗体 325

12.1	用户界面概述	325
12.2	MDI 窗体的构成和创建	326
12.3	MDI 窗体的应用	327

习题	328
----	-----

第 13 章 程序调试与部署 329

13.1	常见错误	329
------	------	-----

13.1	常见错误	329
13.2	规范化编程	331
13.3	Visual Basic 中的调试工具	332
13.3.1	编辑器设置——自动语法检测	332
13.3.2	调试工具	332
13.3.3	程序运行模式	333
13.3.4	添加监视	334
13.4	错误处理	335
13.5	应用程序的部署	336
习题		338
参考文献		339
附录 A 常用工具类库		340
A.1	常用工具类库	340
A.1.1	常用工具类库	340
A.1.2	常用工具类库	340
A.1.3	常用工具类库	340
A.1.4	常用工具类库	340
A.1.5	常用工具类库	340
A.1.6	常用工具类库	340
A.1.7	常用工具类库	340
A.1.8	常用工具类库	340
A.1.9	常用工具类库	340
A.1.10	常用工具类库	340
A.1.11	常用工具类库	340
A.1.12	常用工具类库	340
A.1.13	常用工具类库	340
A.1.14	常用工具类库	340
A.1.15	常用工具类库	340
A.1.16	常用工具类库	340
A.1.17	常用工具类库	340
A.1.18	常用工具类库	340
A.1.19	常用工具类库	340
A.1.20	常用工具类库	340
A.1.21	常用工具类库	340
A.1.22	常用工具类库	340
A.1.23	常用工具类库	340
A.1.24	常用工具类库	340
A.1.25	常用工具类库	340
A.1.26	常用工具类库	340
A.1.27	常用工具类库	340
A.1.28	常用工具类库	340
A.1.29	常用工具类库	340
A.1.30	常用工具类库	340
A.1.31	常用工具类库	340
A.1.32	常用工具类库	340
A.1.33	常用工具类库	340
A.1.34	常用工具类库	340
A.1.35	常用工具类库	340
A.1.36	常用工具类库	340
A.1.37	常用工具类库	340
A.1.38	常用工具类库	340
A.1.39	常用工具类库	340
A.1.40	常用工具类库	340
A.1.41	常用工具类库	340
A.1.42	常用工具类库	340
A.1.43	常用工具类库	340
A.1.44	常用工具类库	340
A.1.45	常用工具类库	340
A.1.46	常用工具类库	340
A.1.47	常用工具类库	340
A.1.48	常用工具类库	340
A.1.49	常用工具类库	340
A.1.50	常用工具类库	340
A.1.51	常用工具类库	340
A.1.52	常用工具类库	340
A.1.53	常用工具类库	340
A.1.54	常用工具类库	340
A.1.55	常用工具类库	340
A.1.56	常用工具类库	340
A.1.57	常用工具类库	340
A.1.58	常用工具类库	340
A.1.59	常用工具类库	340
A.1.60	常用工具类库	340
A.1.61	常用工具类库	340
A.1.62	常用工具类库	340
A.1.63	常用工具类库	340
A.1.64	常用工具类库	340
A.1.65	常用工具类库	340
A.1.66	常用工具类库	340
A.1.67	常用工具类库	340
A.1.68	常用工具类库	340
A.1.69	常用工具类库	340
A.1.70	常用工具类库	340
A.1.71	常用工具类库	340
A.1.72	常用工具类库	340
A.1.73	常用工具类库	340
A.1.74	常用工具类库	340
A.1.75	常用工具类库	340
A.1.76	常用工具类库	340
A.1.77	常用工具类库	340
A.1.78	常用工具类库	340
A.1.79	常用工具类库	340
A.1.80	常用工具类库	340
A.1.81	常用工具类库	340
A.1.82	常用工具类库	340
A.1.83	常用工具类库	340
A.1.84	常用工具类库	340
A.1.85	常用工具类库	340
A.1.86	常用工具类库	340
A.1.87	常用工具类库	340
A.1.88	常用工具类库	340
A.1.89	常用工具类库	340
A.1.90	常用工具类库	340
A.1.91	常用工具类库	340
A.1.92	常用工具类库	340
A.1.93	常用工具类库	340
A.1.94	常用工具类库	340
A.1.95	常用工具类库	340
A.1.96	常用工具类库	340
A.1.97	常用工具类库	340
A.1.98	常用工具类库	340
A.1.99	常用工具类库	340
A.1.100	常用工具类库	340

13.2	规范化编程	331
13.3	Visual Basic 中的调试工具	332
13.3.1	编辑器设置——自动语法检测	332
13.3.2	调试工具	332
13.3.3	程序运行模式	333
13.3.4	添加监视	334
13.4	错误处理	335
13.5	应用程序的部署	336
习题		338
参考文献		339

204 通用财务 1.2.0
205 通用财务 1.2.0
206 通用财务 1.2.0
207 通用财务 1.2.0
208 通用财务 1.2.0
209 通用财务 1.2.0
210 通用财务 1.2.0
211 通用财务 1.2.0
212 通用财务 1.2.0
213 通用财务 1.2.0
214 通用财务 1.2.0
215 通用财务 1.2.0
216 通用财务 1.2.0
217 通用财务 1.2.0
218 通用财务 1.2.0
219 通用财务 1.2.0
220 通用财务 1.2.0
221 通用财务 1.2.0
222 通用财务 1.2.0
223 通用财务 1.2.0
224 通用财务 1.2.0
225 通用财务 1.2.0
226 通用财务 1.2.0
227 通用财务 1.2.0
228 通用财务 1.2.0
229 通用财务 1.2.0
230 通用财务 1.2.0
231 通用财务 1.2.0
232 通用财务 1.2.0
233 通用财务 1.2.0
234 通用财务 1.2.0
235 通用财务 1.2.0
236 通用财务 1.2.0
237 通用财务 1.2.0
238 通用财务 1.2.0
239 通用财务 1.2.0
240 通用财务 1.2.0
241 通用财务 1.2.0
242 通用财务 1.2.0
243 通用财务 1.2.0
244 通用财务 1.2.0
245 通用财务 1.2.0
246 通用财务 1.2.0
247 通用财务 1.2.0
248 通用财务 1.2.0
249 通用财务 1.2.0
250 通用财务 1.2.0
251 通用财务 1.2.0
252 通用财务 1.2.0
253 通用财务 1.2.0
254 通用财务 1.2.0
255 通用财务 1.2.0
256 通用财务 1.2.0
257 通用财务 1.2.0
258 通用财务 1.2.0
259 通用财务 1.2.0
260 通用财务 1.2.0
261 通用财务 1.2.0
262 通用财务 1.2.0
263 通用财务 1.2.0
264 通用财务 1.2.0
265 通用财务 1.2.0
266 通用财务 1.2.0
267 通用财务 1.2.0
268 通用财务 1.2.0
269 通用财务 1.2.0
270 通用财务 1.2.0
271 通用财务 1.2.0
272 通用财务 1.2.0
273 通用财务 1.2.0
274 通用财务 1.2.0
275 通用财务 1.2.0
276 通用财务 1.2.0
277 通用财务 1.2.0
278 通用财务 1.2.0
279 通用财务 1.2.0
280 通用财务 1.2.0
281 通用财务 1.2.0
282 通用财务 1.2.0
283 通用财务 1.2.0
284 通用财务 1.2.0
285 通用财务 1.2.0
286 通用财务 1.2.0
287 通用财务 1.2.0
288 通用财务 1.2.0
289 通用财务 1.2.0
290 通用财务 1.2.0
291 通用财务 1.2.0
292 通用财务 1.2.0
293 通用财务 1.2.0
294 通用财务 1.2.0
295 通用财务 1.2.0
296 通用财务 1.2.0
297 通用财务 1.2.0
298 通用财务 1.2.0
299 通用财务 1.2.0
300 通用财务 1.2.0

第 1 章 Visual Basic 概述

1.1 Visual Basic 6.0 简介

Visual Basic 简称 VB, 是 Microsoft 公司开发的 Windows 应用程序开发工具, 其中 Visual 是一种开发图形用户界面 (GUI) 的方法。Visual Basic 的编程语言是简单易学的 BASIC 语言。

1991 年 Microsoft 公司推出了 Windows 环境下的 Visual Basic1.0 版, 经过不断的改进、完善、升级, 依次推出了 1992 年的 VB2.0 版、1993 年的 VB3.0 版、1995 年的 VB4.0 版、1997 年的 VB5.0 版。1998 年, Microsoft 公司推出了 VB6.0 版, 它不仅继承了之前版本的优点, 而且在开发环境和功能上比之前的版本做了很大的完善和加强。Microsoft 公司 VB6.0 与其他程序开发设计语言 (VC++, VFP, VJ++ 等) 及开发环境打包组成了 Microsoft Visual Studio 98 开发工具套件。

VB 6.0 共有 3 个版本:

- 学习版 (Learning): 是入门版本, 可以方便地建立 Windows 应用程序, 具有建立 Windows 主流应用程序所要的全部工具。
- 专业版 (Professional): 针对职业编程人员, 包括学习版的全部功能, 里边提供了功能齐全的开发工具, 包括 ActiveX 和 Internet 控件开发工具之类的高级特性。
- 企业版 (Enterprise): 最高级的版本, 是针对小组开发环境中建立分布式应用程序的编程人员的版本。企业版除了包括专业版的所有功能外, 还包括一个分布式编程的 Back Office 工具, 用于创建高级、高性能的网络应用程序。

本书介绍的是 Visual Basic 6.0 中文企业版。

1.2 Visual Basic 主要功能和特点

VB 的特点:

(1) 具有面向对象的可视化设计工具

采用传统的程序设计语言编程时, 一般需要通过编写程序来设计应用程序的界面, 在设计过程中看不见界面的实际效果。

VB 采用面向对象程序设计方法 (Object-Oriented Programming), 把程序和数据封装起来作为一个对象, 每个对象都是可视的, 程序员在界面设计的时候可以直接用工具箱在屏幕上“画”出窗口、菜单、命令按键等对象, 并为每个对象设置属性, 程序员仅要对完成事件过程的对象进行编写代码, 因而程序设计的效率可大大提高。

(2) 事件驱动的编程机制

传统的面向过程的程序是由一个主程序和若干个子程序及函数组成的, 程序运行时总是先从主程序开始, 由主程序调用各个子程序和函数, 程序员在编程时必须事先确定整个程序



的执行顺序。

而事件驱动的编程是针对用户触发某个对象的相关事件进行编程,从而达到处理、运算的目的。每个事件都可以驱动一段程序的运行,程序员只要编写响应用户动作的代码,各个动作之间不一定有联系,这样的应用程序代码短,比较容易编写与维护。

(3) 结构化的程序设计语言

VB 具有丰富的数据类型和众多的内部函数,是模块化的结构化程序设计语言,结构清晰、简单,容易学习。

(4) 提供了易学易用的应用程序集成开发环境

VB 为用户设计界面、编写代码、调试程序、编译程序、制作应用程序安装盘等提供了友好的集成开发环境。

除了保持了 VB 的特点,VB6.0 还新增了许多功能和特性:

(1) 数据访问的新特性

VB6.0 利用数据控件可以访问 Access 等多种数据库系统,也可以访问 Excel、Lotus 1_2_3 等多种电子表格。

(2) Internet 功能的增强

VB6.0 提供的 DHML (Dynamic HTML) 设计工具可以使设计者动态地创建和编辑 Web 页面,使用户能开发出多功能的网络应用软件。

(3) 控件、语言和向导方面的新增特性

VB6.0 增加了许多新控件,例如 ADO Data Control、CoolBar Control、DataGrid Control 等,对有些控件增加了功能,如 ImageList Control 等。

VB6.0 中新的语言特性有:用户自定义类型可以作为参数或作为公共属性和方法的返回值,增加了文件系统对象;增加了按名调用等。

VB6.0 向导方面:增强的向导有安装程序向导、数据窗口向导、应用程序向导、类生成工具;新的向导有数据对象生成向导、工具条向导。

(4) 创建 ActiveX 控件更加轻松方便

ActiveX 发展了原来有的 OLE 技术,它使开发人员摆脱了特定语言的束缚,可方便地使用其他应用程序提供的功能,使 VB6.0 能够开发集声音、图像、动画、字处理、电子表格和 Web 等对象一体的应用程序。

此外,VB6.0 还有高度可移植化的代码和在线帮助更加完善等许多新特色。

1.3 Visual Basic 可视化编程环境

英文 Visual 的意思是“视觉的”,“编程环境”指的是用程序设计语言来开发程序的用户界面。VB 编程环境是一种交互的图形窗口式界面,在 VB 的界面提供各种操作,编程人员无需编程就可以完成许多步骤,而且,VB 还提供了大量菜单选项、快捷方式按钮以及各种小窗口,使用起来十分方便。

例如,如果想在窗体上放置一个按钮,可以像在画板上一样,随意点几下鼠标,一个按钮就放好了,这些在以前的编程语言下是要经过相当复杂的工作才能实现。

启动 VB 以后,就进入了 VB 的集成编程环境。这是使用 VB 的第一个画面。它显示了

一个“新建工程”对话框，用户可以选择其中的一个图标建立程序的类型。初学者可以选择默认的“标准 EXE”项，然后单击“打开”按钮，就进入到 VB 开发环境的最重要的编程环境窗口的工作界面了，如图 1-1 所示。

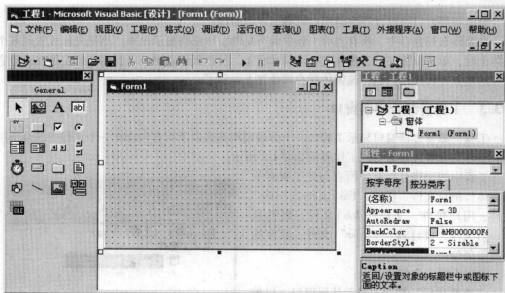


图 1-1 VB 编程环境窗口的工作界面

下面依次介绍 VB 的编程环境窗口的主要内容。

1.3.1 窗体窗口

应用程序可以在窗体窗口中创建窗口、对话框和控件，可在窗体上绘制和查看控件。在设计窗体时，每个窗体窗口都有最大化、最小化和关闭按钮，可创建固定的或可移动的窗体。除非在窗体属性中另有所指，否则，设计出的窗体无论在设计时还是运行时，都将具有相同特征。

在 VB 中，窗体是一个容器，可以使用工具箱中的按钮在窗体上绘制需要的控件。程序运行后窗体窗口的位置可以用“查看”菜单中的“窗体布局窗口”命令在屏幕上预览窗体的布局。

如果在当前工程中再增加新的窗体窗口，方法是：在工程资源管理器窗口中选中当前工程，单击鼠标右键，选择“添加”中的“添加窗体”命令，出现“添

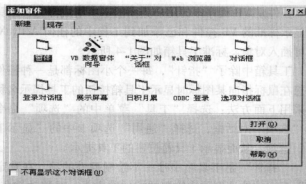


图 1-2 添加窗体的工作界面



加窗体”对话框，使用其中的“新建”选项卡可以添加常用的窗体，“现存”选项卡可以添加现有的窗体。选择了窗体后单击“打开”按钮即可。添加窗体的工作界面如图 1-2 所示。

1.3.2 窗体布局窗口

窗体布局窗口在设计程序时用以可视地定位窗体。

在窗体布局窗口中，所有当前工程中的可见的窗体都将显示出来。当把光标放置到某个窗体上时，单击鼠标左键，窗体变为可移动状态。按住鼠标左键，可以将窗体定位在希望它出现的地方。程序运行后，该窗体就在那个位置上显示了。

1.3.3 工程资源管理器窗口

工程资源管理器窗口显示工程的一个分层结构列表以及所有包含在一个工程中的全部项目。工程资源管理器窗口的工作界面如图 1-3 所示。

工程中的项目可以有：所有与此工程有关的窗体(.frm)文件、模块(.bas)文件、“cls”文件、用户控件、文档(.dob)文件、属性页(.pag)文件、设计器、“.dsr”文件、具有的资源等部分。

窗口上的 3 个按钮如下：

“查看代码”按钮：显示代码窗口，以编写或编辑所选项目的目标代码。

“查看对象”按钮：显示选取的工程，可以是窗体、模块、ActiveX 对象或用户控件的对象窗口。

“切换文件夹”按钮：当正在显示包含在对象文件夹中的个别项目时，可以通过操作此按钮隐藏或显示它们。

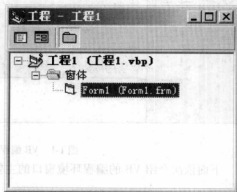


图 1-3 工程资源管理器窗口的工作界面

1.3.4 工具箱

控件是 VB 程序中的用户界面的基本组成部分，工具箱用来显示标准的 VB 控件和已添加到工程中的任何 ActiveX 控件和可插入对象。标准工具箱如图 1-4 所示。

工具箱中除了“指针”，每一个小图标都是一种控件。如果想在鼠标指向某图标时显示工具箱按钮的工具提示名称，可以采用下面方法：选择“工具”菜单中的“选项”命令，进入“选项”对话框，选择“通用”选项卡中的“显示工具提示”选项，从而显示工具箱按钮的工具提示。

“指针”是工具箱中唯一不绘制控件的项。在选定“指针”后只能改变窗体中绘制的控件的大小，或移动这些控件。使用除了“指针”外其他控件时，双击该控件或者单击控件



图 1-4 标准工具箱

后在放置的窗体上拖放。放好的控件也可以用鼠标选中后改变它的大小和位置。

当新建一个工程时,工具箱是标准工具箱状态,上面有常用的控件,如果其中没有需要的控件,可以使用“工程”菜单中的“部件”命令来添加控件。

另外,可以通过定制通用选项卡或专用选项卡将控件进行分类,方法是鼠标放在工具箱上后单击鼠标右键,选择“添加选项卡”。类似的方法可以删除选项卡。

1.3.5 属性窗口

每一个窗体或控件都有自己的属性。属性窗口是用来在设计时显示或改变所选定的窗体或控件属性的窗口。例如,窗体的属性窗口如图 1-5 所示。

属性窗口下面首先是对象框,其中列出当前窗体上所有的对象,可以通过对象框选择当前的对象。对象框列出的对象均具有各自的属性,通过属性列表列出。

属性列表中有两个选项卡,“按字母序”选项卡是按字母顺序列出所选对象的所有属性,在设计时可改变这些对象及其当前的设置。若要改变属性的设定,可以选择属性名然后输入,或直接选取新的设定。“按分类序”选项卡是根据性质列出所选对象的所有属性。例如 BackColor、Caption 以及 ForeColor,都是属于外观的属性。可以折叠这个列表,这样将只看到分类;也可以扩充一个分类,并可以看到其所有的属性。当扩充或折叠列表时,可在分类名称的左边看到一个加号(+)或减号(-)图标。



图 1-5 窗体的属性窗口

属性窗口的最下面是描述窗格,显示属性类型和属性的短描述。

属性窗口的属性修改方法分 3 类:直接修改属性值、选择下拉列表修改值和单击属性右边的对话框按钮通过对话框修改值。

对每个新建的窗体或控件,VB 都自动设置了默认属性值。在设计程序时,可以根据需要在属性窗口改变其中的属性,也可以通过程序的代码修改属性值。

1.3.6 菜单

VB 有两种类型的菜单:内建菜单和快捷方式菜单。

内建菜单出现在 VB 窗口顶端的菜单栏中,每个菜单名称都会有些相应的命令。某些命令具有子菜单,而子菜单又包含一些命令。

举例而言,“格式”菜单包含用来格式化窗体的命令。例如,“视图”菜单上的“工具栏”命令有一个子菜单,它包含工具条的名称及“自定义”命令。可以使用“自定义”命令去修改内建菜单或在菜单栏中添加命令。

快捷方式菜单是一个内含经常使用的命令的菜单,当单击鼠标右键或按 Shift + F10 组合



键时出现。

1.3.7 工具栏

工具栏包含一些常用菜单项的快捷方式按钮。单击某个工具栏按钮,即可执行该按钮所代表的动作。如果希望显示该工具栏按钮的工具提示,可以单击“工具”菜单中“选项”命令,在对话框中的“通用”选项卡中选择“显示工具提示”选项。

VB 工具栏常用的有标准工具栏、编辑工具栏、窗体编辑器、调试工具栏,也可以在菜单中自定义工具栏。需要哪个工具栏,可以单击“视图”菜单中“工具栏”命令,选择对应的工具栏。其中,标准工具栏中包含一些常用菜单项的快捷方式按钮;编辑工具栏中包含代码编辑时频繁用到的一些常用菜单项的快捷方式按钮;调试工具栏中包含在调试代码时频繁用到的一些常用菜单项的快捷方式按钮;窗体编辑器中包含经常用到的对于使用窗体有用的一些菜单项的快捷方式按钮。工具栏界面如图 1-6 所示。

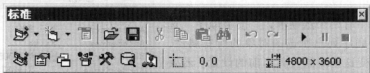


图 1-6 工具栏界面

1.4 Visual Basic 联机帮助

VB 向用户提供了友好的在线帮助和自学功能,为广大读者学习和使用 VB 带来了极大方便。VB 的帮助系统列出了大量的详细信息,不仅可以引导初学者入门,还可帮助各种层次用户完成应用程序的设计。任何一本 VB 教材都有内容的侧重点,很难包括 VB 的全部内容,只有学会使用帮助信息,才能真正全面掌握 VB。

VB 的帮助文件采用的是 MSDN (Microsoft Developer Network) 文档的帮助方式。MSDN Library 中包含了上百个代码示例、文档、技术文章和 Microsoft 开发人员知识库等。

1.4.1 MSDN 的安装

安装 MSDN 的方式有两种:

- 1) MSDN 存放在两张 CD 盘上,可以单独安装。
- 2) 用户也可通过运行第一张 VB 安装盘上的 Setup.exe 程序,通过“用户安装”选项将 MSDN Library 安装到本地机上。

安装完成后,在 VB 中可以直接使用该帮助系统。

在 VB 集成开发环境中,任何时候用户均可以使用上下文相关的在线帮助,它可以根据当前活动窗口或选定的内容来直接定位帮助的内容,是一种最直接、最有效的获得帮助信息的手段。使用的方法是:选定要寻求帮助的内容后按功能键 F1。选定的内容可以是:VB 中的每个窗口;工具箱中的每个控件;窗体或文档内的对象;属性窗口中的某个属性;VB 程序中的关键词(如声明、过程名、函数、属性、方法、事件和特殊对象等)及错误信息。

当对某些内容的帮助需要加深理解时,可单击该帮助处的“示例”超链接,显示出有关的代码示例,用户可以将这些代码复制、粘贴到自己的代码窗口。

1.4.2 MSDN Library 的使用

在 VB 中,选择“帮助”菜单的“内容”、“索引”或“搜索”命令项,即可打开 MSDN Library Visual Studio 6.0 窗口,如图 1-7 所示。其中,“目录”选项卡列出了一个完整的主题的分级列表,通过目录树可查找系统提供的信息;“索引”选项卡可以输入所要搜寻的关键字的头几个字符,帮助系统会自动把相应的关键字找到,并在下面的列表框中显示,选择所要查看的关键字后,单击标签下方的“显示”按钮,则系统提示出相应的帮助信息;“搜索”选项卡可用来通过全文搜索查找信息,输入要查找的单词,单击“列出主题”按钮,则系统显示出相应的帮助信息。



图 1-7 MSDN Library Visual Studio 6.0 窗口

1.4.3 上下文相关帮助

VB 帮助系统支持上下文相关帮助。所谓上下文相关就是指在文本中出现了需要取得帮助的关键字或术语时,或者在窗体中的某一控件具有焦点时,按 F1 键后,帮助系统会弹出帮助窗口,显示关于这些项目的帮助信息。上下文相关意味着不必搜寻“帮助”菜单就能直接获得有关这些部分的帮助。例如,为了获得有关 VB 语言中任何关键词的帮助,只需将插入点置于“代码”窗口中的关键词上并按 F1 键。

在 VB 界面的任何上下文相关部分上按 F1 键,都可显示有关该部分的信息。上下文相关部分如下:

- VB 中的每个窗口(“属性”窗口、“代码”窗口等);
- 工具箱中的控件;
- 窗体或文档对象内的对象;
- “属性”窗口中的属性;
- VB 关键词(声明、函数、属性、方法、事件和特殊对象);
- 错误信息。

一旦打开“帮助”，按 F1 键就可获得怎样使用帮助的信息。

例如，选择某个 VB 窗体，打开属性窗口，选择 Moveable 属性，按 F1 键，帮助窗口出现解释 Moveable 属性的内容。窗体属性帮助窗口的工作界面如图 1-8 所示。

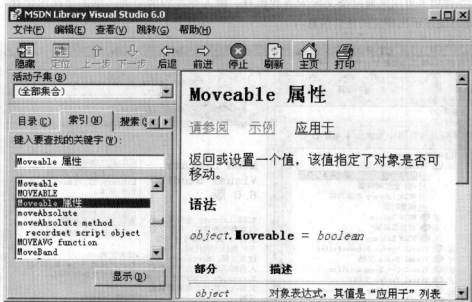


图 1-8 窗体属性帮助窗口的工作界面

1.4.4 运行帮助系统实例

VB 帮助系统提供了很多个程序实例，以便于学习，在 VB 中这些内容默认安装在 C:\Program Files\Microsoft Visual Studio\MSDN98\98VS\2052\SAMPLES\VB98 的子目录中。该子目录下面的每一个子目录中都是一个实例工程包括的各种文件，需要时只要打开其中的工程文件就可以查看其中程序代码和属性，学习其中的使用方法。

1.5 面向对象的基本概念

VB 是一种支持面向对象的程序设计语言，它具有面向对象的特点。

1.5.1 什么是面向对象

面向对象是一种程序设计方法,其基本思想是使用对象、类、继承、封装、消息等基本概念来进行程序设计。

面向对象的程序设计方法的编程技术不同于传统的过程化程序设计,程序设计人员在进行面向对象的程序设计时,不再是单纯地从代码的第一行一直编到最后一行,而是考虑如何创建对象,利用对象来简化程序设计,提供代码的可重用性。对象可以是应用程序的一个自包含组件,一方面具有私有的功能,供自己使用;另一方面又提供公用的功能,供其他用户使用。因此,对象是面向对象的程序设计的基本概念,也是其核心。

VB 是一种面向对象的程序设计语言。它将代码和数据封装集成在一个独立的对象中,当运行这个对象的某个任务时,并不需要知道这个对象是如何工作的,只需要编写一段代码来简单发出一个动作即可。对象、属性、事件和方法都是面向对象程序设计的基本概念,对学习 VB,这种可视化的程序设计工具十分重要。

对象是本课程感兴趣的或要加以研究的事物,是数据与操作相结合的统一体。对象的基本思想是用系统的观点把要研究的事物看成一个整体,整个世界是由各种不同的对象所构成的。

对任何一个对象,可以用对象的属性、事件和方法 3 方面进行描述。

1.5.2 对象的属性

属性是指对象所具有的性质和特征,不同的对象具有不同的属性。正因为如此,各种对象才会有区别。例如,一部电话有一定的颜色和大小,当把一部电话放在办公室中,它又有了一定的位置,而它的听筒也有拿起和挂上两种状态。又如,学生的姓名、性别、身高等,都是学生对象的属性,其中,姓名、性别、身高是属性名,“张三”、“男”、“1.75m”是对应的属性值。

在 VB 中,对象主要分为窗体和控件两类。窗体是用户工作区,所有控件都在窗体中得到了集成,从而构成应用程序的界面;控件是指“空的对象”或基本对象,是应用程序的图形用户界面的一个组件。例如,可以将窗体的属性名称、标题(Caption)、背景颜色(BackColor)等,设置确定的属性值;在该窗体上放置两个按钮控件,这两个按钮控件的属性也可以设置成不同的属性值。

设置对象属性的方法有两种:

- 1) 在程序设计阶段,选择对象后,在属性窗口中直接设置该对象的属性。
- 2) 使用程序代码进行属性赋值。

格式如下:

对象名.属性名 = 属性值

这种方法在程序运行到该语句后,改变属性的值。

例如,一个名称为 Label1 的标签的 Caption 属性为默认值为 Label1,可以用代码改变为“大家好”。代码如下:

```
Label1.Caption = "大家好"
```

在 VB 中,大部分的属性在设计和程序运行时都可用,少数只能在属性窗口设置或者只



能在代码中设置。

第 1 章 1.2.1

1.5.3 对象的事件

每个对象都可以对一个被称为事件的动作进行识别和响应。事件是一种预先定义好的特定动作，由用户或系统激活。在多种情况下，事件是通过用户的交互操作产生的。例如，对一部电话来说，当用户提起听筒时，便激发了一个事件，同样，当用户拨号打电话时也激发了若干事件。

VB 的每一个窗体和控件都有一个预定义的事件集。如果其中某个对象激发了事件，而且，在关联的事件过程中存在代码，则该事件驱动应用程序执行 VB 代码。

在 VB 中，可以激发事件的用户动作包括对某个控件单击鼠标、移动鼠标和按键等。

尽管 VB 中的对象自动识别预定义的事件集，但要判定它们是否响应具体事件以及如何响应具体事件则是编写应用程序的主要工作了。代码部分（即事件过程）与每个事件对应，想让控件响应事件时，就把代码写入这个事件的事件过程之中。

事件过程的一般格式如下：

```
Private Sub 对象名_事件名([参数列表])
```

```
.....' 事件程序代码
```

```
End Sub
```

例如，单击窗体 Form1，使窗体的标题栏变为“你好”，该事件过程代码如下：

```
Private Sub Form1_Click()
```

```
Form1.Caption = "你好"
```

```
End Sub
```

对象所识别的事件类型多种多样，但多数类型为大多数控件所共有。例如，大多数对象都能识别 Click 事件；如果单击窗体，则执行窗体的单击事件过程中的代码；如果单击命令按钮，则执行命令按钮的 Click 事件过程中的代码。每个情况中的实际代码几乎完全不一样。

注意，许多事件伴随其他事件发生，例如在 DblClick 事件发生时，MouseDown、MouseUp 和 Click 事件也会发生。

1.5.4 对象的方法

方法指的是控制对象动作行为的方式。它是对象本身内含的函数或过程，它也是一个动作，是一个简单的不必知道细节的无法改变的事件，但不称做事件；实际上方法是 VB 提供的一些已经封装好的通用子程序，方法也不是随意的，一些对象有一些特定的方法。在 VB 中，方法的调用形式如下：

对象名. 方法名

例如，清除窗体上显示的内容的方法是 Cls，单击窗体 Form1 后调用该方法的代码如下：

```
Private Sub Form1_Click()
```

```
Form1.Cls
```

```
End Sub
```

方法程序是与对象相关联的过程，但又不同于一般的 VB 过程。方法程序紧密地和对象连接在一起，并且与一般 VB 过程的调用方式也有所不同。

事件可以具有与之相关联的方法程序。例如，为 Click 事件编写的方法程序代码将在

Click 事件出现时被执行。方法程序也可以独立于事件而单独存在，此类方法程序必须在代码中被显式地调用。

1.6 一个简单的 Visual Basic 程序

VB 是一个 Windows 应用程序的集成开发环境，用户需要的直观的开发界面、程序的具体设计、代码的调试、应用程序编译成可执行文件直到生成安装程序等步骤都可以在该开发环境中方便地完成。

在安装好 VB 后，启动“开始”菜单，选择“程序”，选择“Microsoft Visual Basic 6.0 中文版”，在弹出菜单中选择“Microsoft Visual Basic 6.0 中文版”即可启动 VB。然后即可创建 VB 应用程序。

创建 VB 应用程序有 3 个主要步骤：

- 1) 创建应用程序界面。
- 2) 设置属性。
- 3) 编写代码。

为了说明这一实现过程，下面学习一个简单的 VB 程序的设计过程。

按照以下步骤创建一个简单应用程序，该应用程序由一个文本框和两个命令按钮组成。单击“命令”按钮，文本框中会出现“Hello, world!”消息，单击“退出”按钮，退出该程序。

下面介绍建立这个简单程序的具体过程。

1.6.1 创建应用程序界面

1. 创建窗体 Form1

首先，建立一个 VB 工程。“工程”是一个管理程序，它包含了一个以“.vbp”为扩展名的工程文件，用来管理用户建立一个应用程序所使用的所有文件，所以首先要建立一个新工程文件。

建立 VB 工程的方法有两种：一种是在启动 VB 开发环境时选择系统默认的标准 EXE 选项，然后单击“打开”按钮；另一种方法是在 VB 开发环境中利用菜单选择“文件”→“新建工程”即可。新建工程的工作界面如图 1-9 所示。

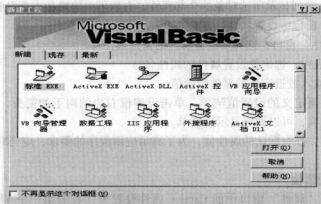




图 1-9 新建工程的工作界面

工程的默认名称是“工程1”，窗体的默认名称是“Form1”。

2. 在窗体 Form1 中添加控件

默认情况下，工程资源管理器、属性窗口、工具箱窗口都显示在开发环境中，如果没有显示，在菜单中找到“视图”项，分别选择“工程资源管理器”、“属性”、“工具箱”，将开发环境的工程资源管理器、属性窗口、工具箱打开。

1) 单击工具箱中的命令按钮图标，鼠标移动到窗体对象窗口后，鼠标指针变成“+”，在窗体中合适的位置按住鼠标左键，拖动鼠标画出命令按钮 Command1 的形状后松开鼠标，用同样办法画出命令按钮 Command2。

2) 单击工具箱中的文本框图标，鼠标移动到窗体对象窗口后，鼠标指针变成“+”，在窗体中合适的位置按住鼠标左键，拖动鼠标画出文本框 Text1 的形状后松开鼠标。

如果对界面中各控件的大小和位置不满意，可以用鼠标进行调整，调整后的窗体如图 1-10 所示。

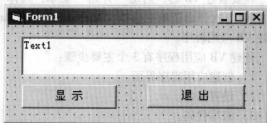



图 1-10 在窗体上加入一个文本框和两个命令按钮

1.6.2 设置对象的属性

VB 中的每一个窗体和控件都有自己的属性。“属性”窗口可以用来显示和设置对象的属性。默认情况下，“属性”窗口已显示在开发环境中，如果没有显示，在菜单中找到“视图”项，选择“属性”，打开“属性”窗口或者单击工具栏上的“属性窗口”按钮，都可以打开“属性”窗口。“属性”窗口包含如下的元素：

- 对象框：显示可设置属性的对象的名字。单击对象框右边的箭头，显示当前窗体的对象列表。
- 排序：从按字母顺序排列的属性列表中进行选取，或从按逻辑（诸如与外观、字体或位置有关的）分类页的层次结构视图中进行选取。
- 属性列表：左列显示所选对象的全部属性，右列可以编辑和查看设置值。

要在“属性”窗口中设置属性，方法如下：

- 1) 打开“属性”窗口，选择要设置属性的窗体或控件。
- 2) 从属性列表中，选定要设置属性的属性名。
- 3) 在右列中输入或选定新的属性设置值。

列举的属性有预定义的设置值清单。单击设置框右边的向下的箭头，可以显示这个清单，或者双击列表项，可以循环显示这清单。

对象建立好后，都有默认的属性值，为了满足应用程序的要求，应当对某些对象的属性值重新进行设置。

1. 设置窗体的属性

选择窗体，将“属性”窗口中的 Caption 属性设置为“我的第一个 VB 程序”，该属性用来设置窗体标题栏的名称。

2. 设置命令按钮的属性

在“属性”窗口的对象框中选 Command1 对象或者直接用鼠标单击选择窗体中的 Command1 按钮,将“属性”窗口中的 Caption 属性设置为“显示内容”,该属性用来设置按钮的名称。同样可以对 Command2 对象的“属性”窗口中的 Caption 属性设置为“退出”。

3. 设置文本框的属性

在“属性”窗口的对象框中选择 Text1 对象或者直接用鼠标单击选择窗体中的 Text1 文本框,将“属性”窗口中的 Text 属性设置为“显示的内容”,该属性用来设置文本框中的显示内容。

1.6.3 编写代码

建立了用户界面并且为其中的对象设置了属性后,就应该为其中的对象编写代码了。代码编辑器窗口是编写应用程序的 VB 代码的地方。进入代码编辑器窗口的方法有 3 种:

- 1) 双击窗体或者其中的控件。
- 2) 在工程资源管理器中,选择“查看代码”项。
- 3) 在菜单中选择“视图”→“代码窗口”项。

代码编辑器窗口包含如下两个元素:

- “对象”列表框:显示所选对象的名称。单击列表框右边的箭头,显示和该窗体有关的所有对象的清单。
- “过程”列表框:列出对象的过程或事件。该框显示选定过程的名称——在目前情况下,是 Click 事件。选取该框右边的箭头可以显示这个对象的全部事件。

编写代码时首先应该在“对象”列表框中选择要编写代码的对象,然后选择该对象在“过程”列表框中的需要编程的某个过程。在程序中需要编写如下代码:

- 1) 编写“显示内容”按钮代码。选择 Command1 对象及该对象的 Click 事件,在代码编辑器窗口中自动列出的过程框架:

```
Private Sub Command1_Click()  
End Sub
```

在这两句中间填写需要的代码:

```
Text1.Text = "大家好"
```

- 2) 编写“退出”按钮代码。同样,在 Command2 对象及该对象的 Click 事件的过程框架中加入结束程序的语句:

```
Private Sub Command2_Click()  
End  
End Sub
```

这样,一个简单的应用程序就编写完成了。

1.6.4 保存文件和打开文件

在界面、属性和代码都设计好之后,应该及时保存文件。

在 VB 中文件类型有很多种,每个应用程序有一个工程文件,每个窗体有一个窗体文件。除此之外还有类模块文件、资源文件等多种不同类型的文件。保存的方法有两种,既可



以为每个应用程序建立一个独立的文件夹，然后将要保存的该应用程序的相关文件都保存在该文件夹中，也可以将工程文件和该工程包括的其他文件保存在不同的文件夹中。在本文的程序中，采用前一种方法保存窗体文件和工程文件。

在 D 盘建立 Myfirstwork 文件夹。

保存窗体文件方法如下：在菜单中选择“文件”，保存 MyfirstForm.frm 的文件。其中 MyfirstForm 为对要保存的窗体取的文件名，“.frm”为窗体文件的扩展名。保存窗体文件的工作界面如图 1-11 所示。

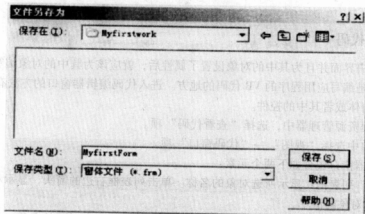


图 1-11 保存窗体文件的工作界面

保存工程文件方法如下：在菜单中选择“文件”→“保存工程”，保存名为 Myfirstwork.vbp 的文件。其中，Myfirstwork 为对要保存的工程取的文件名，“.vbp”为工程文件的扩展名。也可以单击工具栏上的“保存工程”按钮，将工程中包括的各种文件及工程文件本身都进行分别保存。

工程文件就是与该工程有关的全部文件和对象的清单，也是所设置的环境选项方面的信息。每次保存工程时，这些信息都要被更新。工程中的文件全部保存好并关闭该工程之后，如果想重新打开该工程，只要在 VB 打开该工程文件，实际上就打开了该工程文件包括的各个文件。打开工程文件的两种方法如下：

- 1) 双击 Myfirstwork 文件夹中的 Myfirstwork.vbp。
- 2) 在菜单中选择“文件”→“打开工程”项，在现存页中的文件夹中找到要打开的工程或者在最新页中的最新使用过的工程文件中找。打开工程的工作界面如图 1-12 所示。

1.6.5 在 Visual Basic 环境中运行程序

如果要在 VB 环境中运行程序，首先应打开该程序的工程文件，然后单击工具栏中的按钮或者在菜单中选择“运行”→“启动”命令。

运行程序后，单击“显示内容”按钮，将显示如图 1-13 所示的新建的程序的工作界面。单击“退出”按钮，程序结束运行。

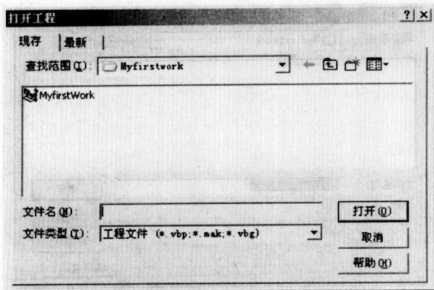


图 1-12 打开工程的工作界面

1.6.6 生成可执行文件

在 VB 中将程序调试正确后，将它编译链接后生成可执行文件，这样，该可执行文件就能够在 VB 开发环境之外运行了。

编译链接后生成可执行文件的方法是，在菜单中选择“文件”→“生成 Myfirstwork.exe”命令，Myfirstwork 是默认的工程文件名，也可以取为其他的可执行文件名。生成可执行文件名的工作界面如图 1-14 所示。

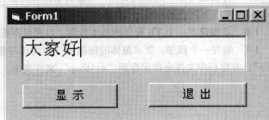


图 1-13 新建的程序的工作界面

这是一个简单 VB 程序的制作和执行的方法，从第 2 章开始，本课程将详细和深入地介绍 VB 程序设计过程中需要的知识。

在对程序设计、编辑、编译和调试、存储后，可以退出 VB 集成开发环境。在 VB 集成开发环境中单击“关闭”按钮或者选择“文件”菜单中的“退出”命令时，VB 会自动判断用户是否修改了工程的内容，并询问用户是保存文件还是直接退出。

习 题

- 1-1 VB 有哪几个版本？各有什么特征？
- 1-2 VB 系统集成环境包括哪几个窗口？各有什么功能？
- 1-3 VB 系统联机帮助怎样安装？联机帮助的使用方法有哪几种？

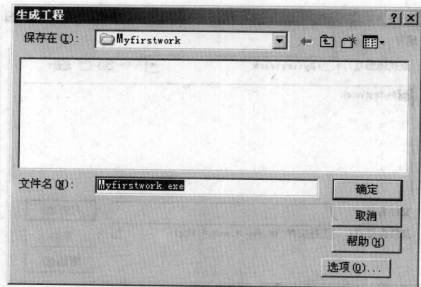


图 1-14 生成可执行文件名的工作界面

- 1-4 什么是对象、对象的属性、对象的事件和对象的方法？
- 1-5 掌握如何启动和退出 VB 系统。
- 1-6 掌握建立一个 VB 程序的基本步骤，以及保存程序和运行程序的方法。
- 1-7 编写一个程序，要求窗体的标题是“我的练习程序”，程序运行后，窗体的标题显示“程序已经运行”，并将程序文件全部保存到“e:\temp”文件夹中。



图 1-14 生成可执行文件名的工作界面

图 1-14 生成可执行文件名的工作界面

图 1-14 生成可执行文件名的工作界面

图 1-14 生成可执行文件名的工作界面

图 1-14 生成可执行文件名的工作界面

图 1-14 生成可执行文件名的工作界面

图 1-14 生成可执行文件名的工作界面

图 1-14 生成可执行文件名的工作界面

图 1-14 生成可执行文件名的工作界面

图 1-14 生成可执行文件名的工作界面

图 1-14 生成可执行文件名的工作界面

第2章 Visual Basic 语言基础

学习一门开发工具, 首先应该掌握它的编程语言, 包括数据类型、常量、变量、运算符和表达式、常用内部函数以及程序的注释和书写规范, 然后才能了解相关规定, 利用这些编程语言进行编程。因此, 本章首先介绍 VB 语言基础。

2.1 数据类型

像大多数编程语言一样, VB 中使用常量和变量来存储值。变量有名字(用来引用该变量所含的值的名词)和数据类型(确定变量可以存储的数据的种类)。不同数据类型的常量和变量在计算机中占有的存储空间是不同的, 从而使得对应的数值的表示范围、精确程度是不同的。所以, 编程人员在学习一种编程语言时, 应该首先了解这种语言能够定义的数据类型有哪些, 每种数据类型在计算机中存储时占用多少空间, 从而知道每种数据类型的表示范围是多少, 当需要对变量定义数据类型时, 才能选择合适的数据类型。

凡是与数据有关的东西就与数据类型有关。

在 VB 中, 数据类型分为基本数据类型和用户定义数据类型两种。

2.1.1 基本数据类型

VB 中定义的基本数据类型分为 Numeric、Byte、String、Boolean、Date、Object 和 Variant 等数据类型。

1. Numeric 数值型

VB 支持几种 Numeric 数据类型。

(1) Integer(整型)

每个 Integer 类型的数据占用 2 个字节的存储空间, 可以存储 -32768 ~ 32767 之间的整数。

(2) Long(长整型)

每个 Long 类型的数据占用 4 个字节的存储空间, 可以存储 -2147483648 ~ 2147483647 之间的整数。

(3) Single(单精度浮点型)

每个 Single 类型的数据占用 4 个字节的存储空间, 可以存储负数时为 -3.402823E38 ~ -1.401298E-45; 正数时为 1.401298E-45 ~ 3.402823E38 之间的数。

(4) Double(双精度浮点型)

每个 Double 类型的数据占用 8 个字节的存储空间, 可以存储负数时为 -1.79769313486232E308 ~ -4.94065645841247E-324; 正数时为 4.94065645841247E-324 ~ 1.79769313486232E308 之间的数。

(5) Currency(货币型)



每个 Currency 类型的数据占用 8 个字节的存储空间, 可以存储 -922337203685477.5808 ~ 922337203685477.5807 之间的整数。

该数据类型在货币计算与定点计算中很有用, 在这种场合精度特别重要。

2. Byte 类型

Byte 类型存储为单精度型、无符号整型、8 位(1 个字节)的数值形式, 范围在 0 ~ 255 之间。

Byte 数据类型在存储二进制数据时很有用。

3. String 类型

包含一连串字符的数据类型, 它们表示字符本身, 而不表示它们的数值。字符串可以包括字母、数字、空白和标点符号。

如果变量总是包含字符串而从不包含数值, 就可将其声明为 String 类型。

字符串有两种: 变长与定长的字符串。

变长字符串最多可包含大约 2G(2^{31}) 个字符。

定长字符串可包含 1 到大约 64K(2^{16}) 个字符。

在 VB 中可用符号(\$) 的类型声明字符来表示一字符串。

4. Boolean 类型

若变量的值只是“true/false”、“yes/no”、“on/off”信息, 则可将它声明为 Boolean 类型。Boolean 的默认值为 False。

5. Date 类型

每个 Date 类型的数据占用 8 个字节的存储空间, 可以存储日期范围从 100 年 1 月 1 日到 9999 年 12 月 31 日, 而时间可以从 0:00:00 到 23:59:59。

6. Object 类型

每个 Object 类型的数据占用 4 个字节的存储空间, 利用 Set 语句, 声明为 Object 的变量可以赋值为任何对象的引用。

7. Variant 类型

Variant 是一种特殊的数据类型, 除了定长 String 数据及用户定义类型外, 可以包含任何种类的数据。

1) 若 Variant 是数字值, 最大可达 Double 的范围。

2) 若 Variant 是字符值, 与变长 String 有相同的范围。

2.1.2 用户定义数据类型

不同类型的变量可以组合起来用来创建用户定义数据类型。用户定义数据类型可包含一个或多个任意数据类型的元素。

用 Dim 语句创建用户定义的数组和其他数据类型。

1. 定义用户定义数据类型

可以用 Type 语句创建用户定义数据类型, 该语句必须置于模块的声明部分。用户定义数据类型可以用适当的关键词声明为 Private 或 Public, 格式如下:

[Public/Private] Type 自定义数据类型

元素一 As 已有的数据类型

元素二 As 已有的数据类型

元素三 As 已有的数据类型

.....

End Type

其中, Private 用于声明只能在包含该声明的模块中使用的用户定义数据类型; Public 用于声明可在所有工程的所有模块的任何过程中使用的用户定义数据类型。

例如,可以在模块的声明段声明一个名为 Student 的模块级自定义数据类型,代码如下:

```
Private Type Student
```

```
    IntNum As Integer
```

```
    StrName As Integer
```

```
    DtmBirthday As Date
```

```
    intMark As Integer
```

```
End Type
```

2. 声明用户定义数据类型的变量

要使用用户定义数据类型,需要声明用户定义数据类型的变量,这时 VB 系统按照该类型分配变量的内存单元,声明用户定义数据类型的变量与声明基本数据类型的方法类似,例如:

```
Dim x,y As Student
```

声明 x 和 y 两个变量,类型为 Student。

3. 用户定义数据类型变量的访问和引用

在 VB 系统中,使用“.”来引用用户定义数据类型变量中的成员项。对这种变量的元素赋值和从元素中引用的方法类似于属性的设置和获取,例如:

```
x.StrName = "张三"
```

```
If x.DtmBirthday > #1/1/92# Then
```

如果两个变量都属于同一个用户定义数据类型,也可以将其中一个变量赋给另一个变量。这种赋值是将一个变量的所有元素赋给另一个变量的对应元素,例如:

```
y = x
```

2.2 常量

VB 程序经常需要对一些文本或数值的内容进行存储,经常会发现用户程序代码中一些数值从不改变,并且一次又一次地反复出现,在这种情况下,可用常量来表示这些数值。例如,将圆周率定义为常量 Pi,在程序中就可以使用 Pi 代替这个常数。另外,常量的处理比变量快,因为程序运行时,常量值不需要查找,编译器只要把常量名换成常数即可,这样保证了程序执行更快。因此,一般来说,程序中能够用常量表示的尽量使用常量表示,这样可以用有意义的符号表示数据,增强程序的可读性。

VB 常量分为直接常量、系统内部定义常量和符号常量(自定义常量)。

2.2.1 直接常量

直接常量是系统提供的可以直接使用的常量,包括数值常量、字符串常量、布尔常量、

日期常量。

1. 数值常量

数值常量共有 5 种表示方式：整数、长整数、定点数、浮点数和字节数。

(1) 整数

十进制整数只能包含数字 0~9、正负号。十进制整型数的范围为 -32768 ~ +32767。例如，-5、12345、0。

十六进制数由数字 0~9、A~F 或 a~f 组成，并以 &H 引导，其后面的数据位数小于等于 4 位，其范围为 &H0 ~ &HFFFF。

八进制数由数字 0~7 组成，并以 &O 或 & 引导，其后面的数据位数小于等于 6 位，其范围为 &O0 ~ &O177777。

(2) 长整数

其数字的组成与整数相同。

十进制长整数的范围为 -2147483648 ~ +2147483647。

十六进制长整数以 &H 开头，以 & 结尾，其范围为 &H0& ~ &HFFFFFFF&。

八进制长整数以 &O 或 & 开头，以 & 结尾，其范围为 &O0& ~ &O3777777777777777&。

(3) 定点数

定点数是带有小数点的正数或负数。定点数表示数的范围比较小。

例如，3.141593、123.45、-100.05、0.0。

定点数可以是单精度也可以是双精度。

(4) 浮点数

浮点数分为单精度浮点数和双精度浮点数。

双精度浮点数存储为 IEEE 64 位(8 个字节)浮点数值的形式，它的范围在负数的时候为 -1.79769313486232E308 ~ -4.94065645841247E-324，而正数的时候为 4.94065645841247E-324 ~ 1.79769313486232E308。双精度浮点数的类型声明字符是数字符号(#)。

单精度浮点数存储为 IEEE 32 位(4 个字节)浮点数值的形式，它的范围在负数的时候为 -3.402823E38 ~ -1.401298E-45，而在正数的时候为 1.401298E-45 ~ 3.402823E38。单精度浮点数的类型声明字符为感叹号(!)。

例如，0.123#，则该数为双精度浮点数，占 8 个字节；0.123!，则该数为单精度浮点数，占 4 个字节。

还可以用科学计数法来表达浮点数，其中指数符号 E 或 D 的含义为乘以 10 的幂次。

例如，1.23E+6(表示 1.23×10^6)、-1.23D-7(表示 -1.23×10^{-7})。

(5) 字节数

字节数是从 0~255 的无符号数，所以不能表示负数。

例如，96、100、0。

2. 字符串常量

字符串常量就是用双引号括起来的一串字符。这些字符可以是除双引号(")和回车、换行符以外的所有字符。

例如，"Visual Basic"、"1234.56"、"True"、"1/9/2005"都是字符串常量。

字符串常量的长度是指字符串中含有的字符个数。

空格也是合法的字符,所以" "是合法的字符串,它的长度是其中的空格个数。

如果一个字符串仅有双引号(即双引号中无任何字符,也不含空格),则称该字符串为空串。例如"",空串中没有任何字符,字符串的长度是零。

如果在字符串中包含双引号,必须使用连续的两个双引号表示一个双引号。例如,"VB 中使用\"号作为字符串的分解符\"。

3. 布尔常量

布尔常量只有 True(真)和 False(假)两个值。

4. 日期常量

用两个“#”符号把表示日期和时间的值括起来表示日期常量。

例如,#12/18/2006#、#8:30:00AM#、#2001.6.1#。

2.2.2 系统内部常量

内部或系统定义的常量是 VB 和控件提供的。这些常量可与应用程序的对象、方法和属性一起使用,在代码中可以直接使用它们。

系统定义的常量位于对象库中,在对象浏览器中的 Visual Basic(VB)和 Visual Basic for Application(VBA)等对象库中列出了 VB 的常量。可以在“对象浏览器”中查看内部常量。方法是打开或新建一个 VB 程序,单击“视图”菜单中的“对象浏览器”命令,则进入对象浏览器窗口,如图 2-1 所示。



图 2-1 对象浏览器窗口

在下拉列表框中选择 VB 或 VBA 对象库,然后在“类”列表框中选择常量组,右侧的成员列表中即显示预定义的常量,窗口底端的文本区域中将显示该常量的功能。

VB 提供了很多内部常量,以方便编程。例如,用来表示真假的布尔型常量“True”和“False”就是比较常用的两个内部常量,在 VB 中,用 -1 代表真(实际上只要是非 0 整数都

可以用来表示真), 0 代表假, 而 VB 已经定义了这两个常量。因此在编程时可以不记真是什么数、假是什么数, 只需要记住比较容易记忆的英语单词: True 和 False 即可。因此下面两个语句是完全等价的:

```
Text1.Enabled = False
```

```
Text1.Enabled = 0
```

使用内部常量的好处不只是易于记忆, 而且当在程序编辑窗口中编写程序代码遇到需要使用内部常量时, VB 会自动列出相应的可供选择的常量。这时, 只要用光键选择相应的常量或输入该常量的前几个字符, VB 会自动选择要输入的常量, 然后按“空格”键或“回车”键, 就可以输入相应的常量了。当输入“=”时, 将会自动弹出两个常量“False”和“True”供选择。若在此时输入了字母“f”, 则常量“False”将会反相显示, 表示要输入的可能是这个常量, 如果这时按下了“空格”键或“回车”键, 常量“False”就会输入到该行中; 同样, 若输入了字母“t”并按下了“空格”键或“回车”键, 则输入的就是常量“True”。

在程序员为属性或方法变量输入数据时, 尽量使用系统已经定义好的常量, 这样可以使编程更加方便, 而且在程序中注重使用常量也是一种良好的编程习惯。

2.2.3 符号常量

符号常量也叫用户自定义常量。

尽管 VB 内部定义了大量的常量, 但是有时用户需要创建自己的符号常量。用户定义常量使用 Const 语句来给常量分配名字、值和类型。

用户定义的符号常量的作用范围有过程级符号常量、模块级符号常量和全局符号常量 3 种。

1. 过程级符号常量

过程是模块的基本组成部分, 是 VB 代码的最小单元。过程级符号常量只能在过程中定义, 它的作用范围只限于定义它的过程之内。也就是说, 一个过程级符号常量只能被定义它的过程使用, 别的过程无权访问该常量。因此, 可以在不同的过程中使用同名的过程级符号常量, 它们是互不影响的。当过程结束时, 过程级符号常量所占用的内存空间就会自动释放。

声明过程级符号常量的格式如下:

Const 常量名[AS 类型名] = 表达式

例如, 新建一个窗体, 上面放两个按钮, 它们的 Click 事件如下:

```
Private Sub Command1_Click()
```

```
Const x = 3
```

```
MsgBox("常量 x = " & CStr(x))
```

```
End Sub
```

```
Private Sub Command2_Click()
```

```
Const x = 2
```

```
MsgBox("常量 x = " & CStr(x))
```

```
End Sub
```

由于 x 是过程级符号常量, 作用范围仅局限于本过程, 所以两个过程中使用同一个常量

名不会产生冲突。

2. 模块级符号常量

模块是构成 VB 工程的一部分，是包含数据和过程的集合。在一个模块中通常包含了多个过程。模块级符号常量为该模块中所有过程所共有，也就是说模块中的所有过程都可以访问属于该模块的模块级符号常量。

添加模块的方法是：“工程”菜单中的“添加模块”命令。

在模块级别中用 Dim 或 Private 声明的常量，对该模块中的所有过程都是可用的，但对于其他模块的代码不可用。

3. 全局符号常量

在一个工程中通常可能包含着多个模块，全局符号常量则是为该工程中所有模块所共有的。有些需要各个模块的不同过程中传递信息，这时就要用到全局符号常量。对于全局符号常量，该工程内的任一过程都可以对其访问。全局符号常量一般是在标准模块的声明部分定义的。

2.2.4 常量的命名

常量需要良好格式的命名约定，这样可以用有意义的符号表示数据，增强程序的可读性。命名规则如下：

- 必须以字母或汉字开头，由字母、汉字、数字或下划线组成，长度不大于 255 个字符。
- 不能使用 VB 中的关键字，并尽量不与 VB 中标准函数名同名，如 Dim、Sin。
- 在同一个范围内必须是唯一的。范围就是可以引用变量的变化域如一个过程、一个窗体等。

通常，常量名的主体是大小写混合的，每个单词的首字母大写。

尽管标准 VB 常量不包含数据类型和范围信息，但是可以使用 g(全局)或 m(模块级)，这样的前缀对于理解一个常量的值和范围还是很有用的。

例如：

mintUserListMax	'对用户列表的最大限制, 整数, 模块级变量
gstrNewLine	'新行字符, 字符串, 应用程序全局使用

2.3 变量

在 VB 应用程序执行时，需要用变量来临时存储数据。变量是由名字和数据类型组成的，其中，变量名是用来引用该变量所含的值的名称，变量类型是确定变量可以存储的数据的种类。也可以将变量看做是数值在内存中存放数值的位置，即内存位置的名称，因此，程序在执行过程中通过变量来存取内存中的数据。

2.3.1 变量的声明

1. 隐式声明

VB 中也允许变量不经过声明就直接使用，这种称为隐式声明，所有隐式声明的变量都

是变体型的。例如：

```
Function SafeSqr(num)
```

```
TempVal = Abs(num)
```

```
SafeSqr = Sqr(TempVal)
```

```
End Function
```

其中，TempVal 变量未进行变量声明就直接使用了。

虽然这种方法很方便，但是如果把变量名拼错了的话，会导致一个难以查找的错误。例如，假定写了这样一个函数：

```
Function SafeSqr(num)
```

```
TempVal = Abs(num)
```

```
SafeSqr = Sqr(TempVal)
```

```
End Function
```

乍看起来，这两段代码好像是一样的。但是因为在倒数第二行把 TempVal 变量名写错了，所以函数总是返回 0。当 VB 遇到新名字，它分辨不出这是隐式声明了一个新变量，还是仅仅把一个现有变量名写错了，于是只好用这个名字再创建一个新变量。

隐式声明容易造成错误，为了调试程序方便，一般对使用的变量都进行声明，可以在通用声明段使用 Option Explicit 语句来强制显式声明所有变量。

2. 显式声明

所谓显式声明，是指每个变量必须事先做声明，才能够正常使用，否则会出现错误警告。用 Dim 语句进行显式声明格式如下：

Dim 变量名 [As 数据类型]

例如：

```
Dim intX As Integer
```

如果没有“As 类型”，则系统默认为变体类型。

1) 一条语句可以同时定义多个变量，但每个变量必须有自己的类型声明，类型声明不能共用，例如：

```
Dim x,y As Integer '是正确的
```

2) 类型声明字符附加到变量名上的字符，指出变量的数据类型来代替 As 类型，VB 常用类型声明字符见表 2-1。

表 2-1 VB 常用类型声明字符

数据类型	类型声明字符	数据类型	类型声明字符
Double	#	String	\$
integer	%	Single	!
Currency	@	Long	&

例如，Dim intX% 显式声明变量整型变量 intX。

显式声明变量可以避免写错变量名引起的麻烦，要显式声明变量，有两种方法。

1) 在类模块、窗体模块或标准模块的声明段中加入下列语句：

```
Option Explicit
```

2) 在“工具”菜单中选取“选项”，单击“编辑器”选项卡，再复选“要求变量声明”选项。这样就在任何新模块中自动插入 Option Explicit 语句，但不会在已经建立起来的模块中自动插入；所以在工程内部，只能用手工方法向现有模块添加 Option Explicit。

上面的例子中，如果对包含 SafeSqr 函数的窗体或标准模块执行该语句，那么 VB 将认定 TempVal 和 TemVal 都是未经声明变量，并为两者都发出错误信息。随后就可以显式声明 TempVal 变为

```
Function SafeSqr(num)
    Dim TempVal
    TempVal = Abs(num)
    SafeSqr = Sqr(TempVal)
End Function
```

因为 VB 对拼错了的 TemVal 显示错误信息，所以能够立刻明白是什么问题。由于 Option Explicit 语句有助于抓住这些类型的错误，所以最好在所有代码中使用它。

2.3.2 变量的存取

对变量常用的存取方法是使用赋值语句。

赋值语句的格式如下：

[Let] 变量名或属性名 = 表达式

功能是计算表达式的值，然后将表达式的值赋给变量或属性。其中 Let 关键字是可选的。也可以省略该关键字。

下面都是有效的赋值语句：

```
Let x = 3
x = 3
m = m + 1
```

注意：如果两个变量要交换数值，需要使用另外一个临时变量。

例如交换 X 和 Y 两个变量的值：

```
Temp = X
X = Y
Y = Temp
```

2.3.3 变量的作用域

前面用 Dim 语句进行显式声明时对声明的变量名和它的数据类型进行了声明。其实，一个完整的变量的声明应该包括：声明的变量名、变量数据类型、变量作用域和生存期等内容。

声明变量的语句由于所处的位置不同，它的作用范围也不同。与用户定义的符号常量的作用范围一样，根据变量的作用范围分为过程级变量、模块级变量、全局变量 3 种。

1. 过程级变量

在过程内部声明的变量叫过程级变量。别的过程无权访问该变量。因此，可以在不同的过程中使用同名的过程级变量，它们是互不影响的。声明过程级变量的格式有两种。

1) 格式 1:

Dim 变量名 As 数据类型名

2) 格式 2:

Static 变量名 As 数据类型名

其中, Dim 或 Static 为 VB 的关键字。

功能: 定义一个变量为指定的数据类型。这时, 系统为该变量分配存储单元。

这两个语句的区别是: 用 Dim 声明的变量在程序运行进入该声明变量的过程时, 系统为该变量分配存储单元, 当退出该过程时, 系统释放存储单元, 下一次再执行这个过程时, 系统重新为该变量分配存储单元; 用 Static 声明的变量称为静态变量, 它的特点是在程序运行第一次进入该声明变量的过程时, 系统为该变量分配存储单元, 当退出该过程时, 保留该存储单元, 变量值不变, 下一次再执行这个过程时, 该变量值可以继续使用或者重新赋值。

例如:

```
Dim IntVar1 As Integer
```

```
Static IntVar1 As Integer
```

2. 模块级变量

在模块开始部分以 Dim 或 Private 声明的变量叫模块级变量。模块级变量在该模块中的所有过程都是可用的。但对其他模块的代码不可用。声明模块级变量的格式有两种。

1) 格式 1:

Dim 变量名 As 数据类型名

2) 格式 2:

Private 变量名 As 数据类型名

其中, Dim 或 Private 为 VB 中的关键字。

在模块中定义的模块级变量这两种格式是等价的。例如:

```
Private IntVar1 As Integer
```

```
Dim IntVar1 As Integer
```

两条语句都是在模块中定义一个在模块内都可使用的名为 IntVar1 的变量。

3. 全局变量

在标准模块内以 Public 声明的变量叫全局变量。全局变量在该程序中的所有过程都是可用的。声明全局变量的格式如下:

Public 变量名 As 数据类型名

其中, Public 为 VB 中的关键字。

例如:

```
Public IntVar1 As Integer
```

2.3.4 变量名

VB 对变量的命名要求与常量命名要求相同, 见 2.2.4 节。

下面是正确的变量名:

X, A1, Sum, My_name

下面是不正确的变量名:

1B, Integer, b[1]

VB 中不区分变量的大小写, 一般变量首字母用大写, 其余用小写; 例如, Mystring 和 mystring 表示的是同一个变量。

一般建议:

- 变量名设计时最好让变量名能够表示它的功能, 例如, Min 表示最小值, Sum 表示和。
- 对变量名较长的可以使用下划线进行分割, 从而增加可读性。
- 为了增加程序的可读性, 可在变量名前加一个缩写的前缀来表明该变量的数据类型和作用范围。并且变量应该总是被定义在尽可能小的范围内。

VB 对变量数据类型的前缀约定见表 2-2。

表 2-2 VB 对变量数据类型的前缀约定

数据类型	前缀	数据类型	前缀
Boolean	Bin	Long	Lng
Byte	Byt	Object	Obj
Currency	Cur	Single	Sng
Date	Dtm	String	Str
Double	Dbl	Variant	Vnt
Integer	Int	Use-Define Type (用户定义类型)	Udt

VB 对变量作用范围的前缀约定如下:

- 全局前缀为: g。
- 模块级前缀为: m。
- 过程前缀为: 无。

例如, gstrUserName 定义的是全局的字符串变量名。

2.3.5 变量的数据类型转换

因为不同数据类型的数据在计算机内存中占用的空间不同, 使表示数的范围和精度也不相同, 当不同数据类型的数据进行运算或赋值时, 需要转换变量的数据类型。类型转换的方法有两种: 隐式转换和显式转换。

1. 隐式转换

“隐式转换”不需要源代码中的任何特殊语法。在下面的示例中, 在将 k 的值赋给 q 之前, VB 将该值隐式转换成单精度浮点值:

```
Dim k As Integer
Dim q As Double
k = 432
q = K
```

2. 显式转换

“显式转换”使用类型转换关键字。VB 提供了几个这样的关键字, 它们将括号中的表达式强制转换为所需的数据类型。VB 常用类型转换函数见表 2-3。

表 2-3 VB 常用类型转换函数

函 数	功 能	expression 参数范围
CBool	转换为 Boolean 类型	任何有效的字符串或数值表达式
CByte	转换为 Byte 类型	0 ~ 255
CCur	转换为 Currency 类型	-922337203685477.5808 ~ 922337203685477.5807
CDate	转换为 Date 类型	任何有效的日期表达式
CDbl	转换为 Double 类型	负数为 -1.79769313486232E308 ~ -4.94065645841247E-324; 正数为 4.94065645841247E-324 ~ 1.79769313486232E308
CDec	转换为 Decimal 类型	零变比数值, 即无小数位数值, 为 +/- 79228162514264337593543950335。对于 28 位小数的数值, 范围则为 +/- 7.9228162514264337593543950335; 最小的可能非零值是 0.00000000000000000000000000000001
CInt	转换为 Integer 类型	-32768 ~ 32767, 小数部分四舍五入
CLng	转换为 Long 类型	-2147483648 ~ 2147483647, 小数部分四舍五入
CSng	转换为 Single 类型	负数为 -3.402823E38 ~ -1.401298E-45; 正数为 1.401298E-45 ~ 3.402823E38
CStr	转换为 String 类型	依据 expression 参数返回 Cstr
CVar	转换为 Variant 类型	若为数值, 则范围与 Double 相同; 若不为数值, 则范围与 String 相同

例如:

```

1 Dim MyDouble, MyString
2 MyDouble = 437.324 'MyDouble 为 Double 类型
3 MyString = CStr(MyDouble) 'MyString 的内容为"437.324"

```

将 MyDouble 用函数转换成字符串后, 赋值给 MyString。

2.4 运算符和表达式

2.4.1 运算符

运算符是代表 VB 某种运算功能的符号。VB 程序会按运算符的含义和运算规则执行实际的运算操作。VB 中的运算符包括赋值运算符、数学运算符、比较运算符、逻辑运算符和连接运算符。

1. 赋值运算符

VB 中的赋值运算符用来给变量、变长数组或对象的属性赋值, 即把运算符右边的内容赋给运算符左边的变量或属性。VB 中的赋值运算符是“=”, 其一般格式如下:

变量 = 值

其中, “变量”可以是变量、数组的元素或属性。“值”可以是常量、变量、表达式或函数返回值。

例如:

```
Dim IntX As Integer
```

```
IntX = IntX + 1
```

或者

Command1.Caption = "确定" 将 Command1 的 Caption 属性赋值为确定

2. 数学运算符

用来进行数学计算的运算符称为数学运算符。

语法格式如下：

计算结果 = 表达式 1 数学运算符 表达式 2

其中，除 + 运算符外，表达式 1 和表达式 2 都应该是数值表达式。

运算符：用来求一个数字的某次方。

如果数学表达式 1 或数学表达式 2 为 Null 表达式，则 Result 值也为 Null。

例如：

```
Dim MyValue
MyValue = 2^2      '返回 4
MyValue = (-5)^3   '返回 -125
```

* 运算符：用来将两数相乘。

如果一个或两个表达式为 Null 表达式，Result 为 Null。如果一个表达式为 Empty，则按 0 处理。

例如：

```
Dim MyValue
MyValue = 2 * 2     '返回 4
MyValue = 459.35 * 334.90 '返回 153836.315
```

/ 运算符：用来进行两个数的除法运算并返回一个浮点数。

如果一个或两个表达式为 Null 表达式，Result 为 Null。任何表达式为 Empty 时，则按 0 处理。

例如：

```
Dim MyValue
MyValue = 10/4      '返回 2.5
MyValue = 10/3      '返回 3.333333
```

\ 运算符：用来对两个数作除法并返回一个整数。

在除法操作前，数值表达式四舍五入为 Byte、Integer 或 Long 子类型表达式。

如果任何表达式为 Null，Result 也是 Null。任何表达式为 Empty 时，则按 0 处理。

```
Dim YourObject, MyObject, MyStr
Set MyObject = YourObject  '对象引用赋值, MyObject 和 YourObject 引用同一个对象
YourObject.Text = "Hello World" '初始化属性
MyStr = MyObject.Text      '返回"Hello World", 脱离关联. MyObject 不再引用 YourObject
Set MyObject = Nothing     '释放该对象
```

注意：如果结果不是整数，结果小数部分被删除。

例如：

```
Dim MyValue
MyValue = 11\4      '返回 2
```

MyValue = 9 \ 3 '返回 3

MyValue = 100 \ 3 '返回 33

Mod 运算符：用来对两个数作除法并且只返回余数。

如果任一表达式为 Null，则 Result 也为 Null。任一表达式为 Empty 时按 0 来处理。

例如：

Dim MyResult

MyResult = 10 Mod 5 '返回 0

MyResult = 10 Mod 3 '返回 1

MyResult = 12 Mod 4, 3 '返回 0

MyResult = 12.6 Mod 5 '返回 3

+ 运算符：用来求两数之和。

语法格式如下：

计算结果 = 表达式 1 数学运算符 表达式 2

当两个表达式都是数值时，则相加；当两个表达式都是字符串时，则连接；当一个表达式是数值，另一个表达式是字符串时，则相加。

如果一个表达式或两个表达式都为 Null 表达式，则 Result 为 Null。

如果两个表达式都为 Empty，则 Result 为 Integer 子类型。但是如果一个表达式为 Empty，则返回另一个表达式作为 Result。

例如：

Dim MyNumber, Var1, Var2

MyNumber = 2 + 2 '返回 4

MyNumber = 4257.04 + 98112 '返回 102369.04

- 运算符：用来求两数之差或表示数值表达式的负值。

语法格式如下：

语法格式 1：

result = number1 - number2

语法格式 2：

-number

在语法格式 1 中，- 运算符是用于计算两个数值差值的算术减法运算符。在语法格式 2 中，- 运算符用作单目求反运算符，表示表达式的负数。

如果一个或两个表达式都是 Null 表达式，则 Result 为 Null。如果某个表达式为 Empty，则按 0 值处理。

例如：

Dim MyResult

MyResult = -2 '返回 -2

MyResult = 459.35 - 334.90 '返回 124.45

3. 比较运算符

比较运算符用来比较表达式，是对两个表达式进行两个逻辑量的比较，结果为逻辑型值。

关系运算符有小于(<)、小于等于(<=)、大于(>)、大于等于(>=)、不等(<>)和

等于(=)。关系运算符运算结果见表 2-4。

表 2-4 关系运算符运算结果

运 算 符	结果为 True	结果为 False	结果为 Null
< (小于)	expression1 < expression2	expression1 >= expression2	expression1 or expression2 = Null
<= (小于或等于)	expression1 <= expression2	expression1 > expression2	expression1 or expression2 = Null
> (大于)	expression1 > expression2	expression1 <= expression2	expression1 or expression2 = Null
>= (大于或等于)	expression1 >= expression2	expression1 < expression2	expression1 or expression2 = Null
= (等于)	expression1 = expression2	expression1 <> expression2	expression1 or expression2 = Null
<> (不等于)	expression1 <> expression2	expression1 = expression2	expression1 or expression2 = Null

例如:

```
Dim MyResult, Var1, Var2
MyResult = (45 < 35) '返回 False
MyResult = (45 = 45) '返回 True
MyResult = (4 <> 3) '返回 True
MyResult = ("5" > "4") '返回 True
```

```
Var1 = "5"; Var2 = 4 '设置变量初值
MyResult = (Var1 > Var2) '返回 True
```

```
Var1 = 5; Var2 = Empty
MyResult = (Var1 > Var2) '返回 True
```

```
Var1 = 0; Var2 = Empty
MyResult = (Var1 = Var2) '返回 True
```

其他运算符还有 Is 和 Like。格式如下:

Result = expression1 关系运算符 expression2

例如:

```
Result = object1 Is object2
Result = string Like pattern
```

Is: 用于对象型数据, 相当于适用于一般数据的 "=", 用来比较两个对象变量, 如果它们引用的是同一个对象, 则运算结果为 True, 否则为 False。

格式如下:

Result = object1 Is object2

例如:

```
Dim MyObject, YourObject, ThisObject, OtherObject, ThatObject, MyCheck
Set YourObject = MyObject '指定对象引用
Set ThisObject = MyObject
Set ThatObject = OtherObject
MyCheck = YourObject Is ThisObject '返回 True
```

MyCheck = ThatObject Is ThisObject '返回 False

'假设 MyObject <> OtherObject

MyCheck = MyObject Is ThatObject '返回 False

result = string Like pattern

Like: 用于字符型数据, 功能是, 左边的字符串与右边含有通配符的字符串是否匹配? 如果匹配, 结果为 True, 如果不匹配, 结果为 False。

例如, 验证字符串是否是电话号码或者社会保险号码, 可以使用 Like 运算符来判断。

Like 运算符的通配符见表 2-5。

表 2-5 Like 运算符的通配符

通 配 符	含 义	通 配 符	含 义
?	任何单一字符	[! charlist]	不在 charlist 中的任何单一字符
*	零个或多个字符	[char1-char2]	char1-char2 中的任何单一字符
#	任何一个数字(0~9)	[! char1-char2]	不在 char1-char2 中的任何单一字符
[charlist]	charlist 中的任何单一字符		

例如:

```
If (800)555-5555 "Like "(###)###-####" Then
```

'注释:有效的电话号码

```
End If
```

```
If "1ABCDEF" Like "#[A-Z]*" Then
```

'注释:有效

```
End If
```

```
If "1ABCDEF" Like "#[a-z]*" Then
```

'注释:有效

```
End If
```

Like 运算符可以简化验证有效性逻辑并提高易读性。

4. 逻辑运算符

逻辑运算符用来进行两个逻辑量的运算, 结果为逻辑值。

除了 Not 运算符外的语法格式如下:

[结果表达式] = 表达式1 逻辑运算符 表达式2

其中, 结果表达式是可选的。

And 运算符: 用来对两个表达式进行逻辑连接。

如果两个表达式的值都是 True, 则 Result 是 True。如果其中一个表达式的值是 False, 则 Result 是 False。

例如:

```
Dim A,B,C,D,MyCheck
```

```
A = 10; B = 8; C = 6; D = Null '设置变量初值
```

```
MyCheck = A > B And B > C '返回 True
```

```
MyCheck = B > A And B > C '返回 False
```

```
MyCheck = A > B And B > D '返回 Null
```

Eqv 运算符：用来对两个表达式进行逻辑等价运算。

如果两个表达式的值都是 True 或 False，则 Result 是 True。如果有一个表达式是 Null，则 Result 也是 Null。例如：

```
Dim A, B, C, D, MyCheck
```

```
A = 10; B = 8; C = 6; D = Null '设置变量初值
```

```
MyCheck = A > B Eqv B > C '返回 True
```

```
MyCheck = B > A Eqv B > C '返回 False
```

```
MyCheck = A > B Eqv B > D '返回 Null
```

Not 运算符：用来对表达式进行逻辑否定运算。

语法格式如下：

[结果表达式] Not 运算符 表达式

如果表达式 True 结果为 False；表达式是 False 结果为 True；表达式是 Null 结果为 Null。

例如：

```
Dim A, B, C, D, MyCheck
```

```
A = 10; B = 8; C = 6; D = Null '设置变量初值
```

```
MyCheck = Not(A > B) '返回 False
```

```
MyCheck = Not(B > A) '返回 True
```

```
MyCheck = Not(C > D) '返回 Null
```

Or 运算符：用来对两个表达式进行逻辑析取运算。

如果两个表达式中至少有一个为 True，则 Result 为 True；如果两个表达式都是 False，结果为 False；其余情况，结果为 Null。

例如：

```
Dim A, B, C, D, MyCheck
```

```
A = 10; B = 8; C = 6; D = Null '设置变量初值
```

```
MyCheck = A > B Or B > C '返回 True
```

```
MyCheck = B > A Or B > C '返回 True
```

```
MyCheck = A > B Or B > D '返回 True
```

```
MyCheck = B > D Or B > A '返回 Null
```

Xor 运算符：用来对两个表达式进行逻辑互斥或运算。

如果表达式中有一个而且只有一个值为 True，则 Result 为 True。但是，如果表达式中有一个为 Null，则 Result 也为 Null。

例如：

```
Dim A, B, C, D, MyCheck
```

```
A = 10; B = 8; C = 6; D = Null '设置变量初值
```

```
MyCheck = A > B Xor B > C '返回 False
```

```
MyCheck = B > A Xor B > C '返回 True
```

```
MyCheck = B > A Xor C > B '返回 False
```

```
MyCheck = B > D Xor A > B '返回 Null
```

Imp 运算符：用来对两个表达式进行逻辑蕴涵运算。

表达式 1 为真, 表达式 2 为假时, 结果才为假, 其余都为真。功能与 (Not A) Or B 等价。

例如:

```
Dim A, B, C, D, MyCheck
A = 10; B = 8; C = 6; D = Null '设置变量初值
MyCheck = A > B Imp B > C '返回 True
MyCheck = A > B Imp C > B '返回 False
MyCheck = B > A Imp C > B '返回 True
MyCheck = B > A Imp C > D '返回 True
```

5. 连接运算符

& 运算符: 用来强制两个表达式作字符串连接。

例如:

```
Dim MyStr
MyStr = "Hello" & "World" '返回"Hello World"
MyStr = "Check" & 123 & "Check" '返回"Check 123 Check"
```

+ 运算符: 也可以用来作字符串的串接操作。

例如:

```
Dim MyNumber, Var1, Var2
Var1 = "34"; Var2 = "6" '用字符串初始化混合变量的值
MyNumber = Var1 + Var2 '返回"346"(字符串被串接起来)
```

2.4.2 表达式

计算机中, 对数据的加工是通过运算(操作)进行的。运算可以通过由操作符(运算符)和操作数(操作对象)组成的表达式来完成。表达式描述了对哪些数据, 以何种顺序进行什么样的操作。操作数可以是变量、常量及函数, 甚至对象属性。VB 中提供了丰富的运算符, 可以构成多种表达式, 完成许多复杂运算和操作。

1. 表达式的组成

表达式由常量、变量、运算符、函数和圆括号按一定的规则组成, 通过运算后有一个结果, 运算结果的类型由数据和运算符共同决定。

2. 表达式的书写规则

- 乘号不能省略;
- 括号必须成对出现, 均使用圆括号, 可以嵌套, 但必须配对;
- 表达式从左到右在同一基准上书写, 无高低、大小之分。

例如:

```
sqr((3 * x + y) - z) / (x * y) ^ 4
```

3. 不同数据类型的转换

表达式中操作数的数据类型应该符合要求, 不同的数据应该转换成同一类型。在算术运算中, 如果操作数的数据精度不同, VB 规定运算结果采用精度较高的数据类型。

在 VB 中允许整型、实型、双精度型、货币型等不同类型的数据进行混合运算。

例如:

10 + 1.5 - 8765.1234

在不同类型的运算对象进行运算时先要转换成同一类型，然后再进行运算，结果的类型一般为两个运算对象中存储长度较长的那个对象的类型。

2.4.3 运算符的优先级

一个表达式中进行若干操作时，每一部分都会按预先确定的顺序进行计算求解，称这个顺序为运算符的优先级。

在一个表达式中，优先级高的运算符先计算，优先级低的运算符后计算，并且 VB 系统中运算符采用左结合，即相同优先级的运算符按照从左到右顺序计算。

从大的运算符看，优先级按算术运算符、字符串运算符、关系运算符、逻辑运算符依次降低。

对算术运算符，优先级按指数运算(^)、负数(-)、乘法和除法(*、/)、整数除法(\)、求模运算(Mod)、加法和减法(+、-)依次降低。

对逻辑运算符，优先级按 Not、And、Or、Xor、Eqv、Imp 依次降低。

关系运算符中的运算符优先级相同。

运算符的优先级见表 2-6 所示。

表 2-6 运算符的优先级

运算符类别	运 算 符
算术运算符	指数运算(^)
	负数(-)
	乘法和除法(*、/)
	整数除法(\)
	求模运算(Mod)
	加法和减法(+、-)
字符串运算符	字符串连接(&)
关系运算符	相等(=)、不等(<>)、小于(<)、大于(>)、小于或相等(<=)、大于或相等(>=)、Like、Is
逻辑运算符	Not
	And
	Or
	Xor
	Eqv
	Imp

如果在多种运算符的表达式，想人为改变运算次序，必须通过圆括号“()”。

例如，要计算 $\frac{a+b}{a-b}$ 的表达式应为

$$(a+b)/(a-b)$$

注意：对于存在多种运算符的表达式，可增加圆括号使表达式更清晰。

再如,编写一个表达式,判断x是否是闰年,如果是,表达式的值为True,如果不是,表达式的值为False。

判断闰年的条件必须符合以下两个规则之一:

- 1) 能被4整除,但不能被100整除。
- 2) 能被400整除。

用逻辑表达式来表示,该表达式可以写为

$$\text{Year Mod } 4 = 0 \text{ And Year Mod } 100 <> 0 \text{ Or Year Mod } 400 = 0$$

也可以写为

$$((\text{Year Mod } 4) = 0 \text{ And } (\text{Year Mod } 100) <> 0) \text{ Or } (\text{Year Mod } 400) = 0$$

显然,后一种方法程序的可读性更强。

2.5 常用的内部函数

利用VB开发环境进行编程的一个重要方面是函数的编写和使用。VB系统中的函数分为外部函数和内部函数。外部函数是编程人员自己编写的。内部函数是VB系统提供的。

每个内部函数完成某个特定的功能,使用内部函数之前必须知道它的名称和相应的要求,尽管内部函数大家看不到程序的代码,但仍然可以正常使用它。

VB是一个功能强大的应用程序开发工具,其中所包含的丰富的内部函数更是为使用提供了很大的帮助。但是,任何人都不能记住这么多的内部函数,下面选择一些常用的函数给大家进行介绍,希望大家能掌握这些常用内部函数的功能及使用。对不常用内部函数,大家可以利用帮助文件来了解和使用。

VB开发环境给用户提供的内部函数分几类,有利于程序接受信息的InputBox和显示信息的MsgBox函数,有类型转换函数、数学函数、字符串函数、时间/日期函数、随机函数等。

通常在使用函数时,需要给函数传递一个或多个值,传递给函数的值叫参数。

函数调用方法如下:

函数名(参数列表) '有参函数

函数名 '无参函数

说明:

- 1) 使用内部函数要注意参数的个数及其参数的数据类型。
- 2) 注意参数列表的顺序。参数列表中,如果是必需的参数,必须列出;可选的参数,如果需要可以列出,不需要则可以不列出。如果要省略某些位置参数,则必须加入相应的逗号分界符。

- 3) 要注意函数的定义域(自变量或参数的取值范围)。

例如, $\text{sqr}(x)$, 要求: $x \geq 0$ 。

- 4) 要注意函数的值域。

例如, $\text{exp}(23999)$ 的值就超出实数在计算机中的表示范围。

2.5.1 输入函数 InputBox

InputBox 函数的功能是: 用户从键盘输入的内容可以作为函数的返回值返回到当前的程

序中去。

InputBox 函数使用的是对话框界面,方便程序获得用户的输入内容。

函数格式如下:

InputBox (prompt[,title][,default][,xpos][,ypos][,helpfile,context])

参数说明:

prompt:必需的,作为对话框消息出现的字符串表达式。prompt 的最大长度大约是 1024 个字符,由所用字符的宽度决定。如果 prompt 包含多个行,则可在各行之间用回车符(Chr(13))、换行符(Chr(10))或回车换行符的组合(Chr(13)& Chr(10))来分隔。

title:可选的,显示对话框标题栏中的字符串表达式。如果省略 title,则把应用程序名放入标题栏中。

default:可选的,显示文本框中的字符串表达式,在没有其他输入时作为默认值。如果省略 default,则文本框为空。

xpos:可选的,数值表达式,成对出现,指定对话框的左边与屏幕左边的水平距离。如果省略 xpos,则对话框会在水平方向居中。

ypos:可选的,数值表达式,成对出现,指定对话框的上边与屏幕上边的距离。如果省略 ypos,则对话框被放置在屏幕垂直方向距下边大约 1/3 的位置。

helpfile:可选的,字符串表达式,识别帮助文件,用该文件为对话框提供上下文相关的帮助。如果已提供 helpfile,则也必须提供 context。

context:可选的,数值表达式,由帮助文件的作者指定给某个帮助主题的帮助上下文编号。如果已提供 context,则也必须提供 helpfile。

InputBox 函数返回值:如果用户单击 InputBox 对话框界面中的“确定”按钮,此函数返回一个输入的字符串,如果单击“取消”按钮,则此函数返回一个长度为零的字符串("")。

例如,编写一个程序,要求用户输入姓名,然后进行判断,如果是管理员“abc”,则在文本框中 Text1 显示内容“欢迎使用本功能!”,如果不是管理员“abc”,则弹出警告提示“您无权使用该功能!”。

```
Dim MyValue As String
```

```
MyValue = InputBox("请输入姓名","显示内容前检查")
```

```
If MyValue <> "abc" Then
```

```
    Text1.Text = "您无权使用该功能!"
```

```
Else
```

```
    Text1.Text = "欢迎使用本功能!"
```

```
End If
```

程序执行到 InputBox 函数语句时,弹出如图 2-2 所示对话框。

如果输入姓名后,单击“确定”按钮,输入内容赋给 MyValue,如果单击“取消”按钮,MyValue 为空串。

如果需要使用帮助文件对操作的内容进行解释,要同时使用 Helpfile 和 Context。

例如,如果要求用户输入数,计算该数的开方数,默认的数是 100,可以提供以下帮助信息:

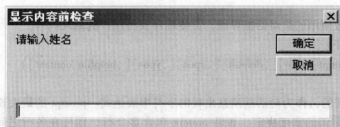


图 2-2 “显示内容前检查”对话框

```
Dim MyValue As String
```

```
MyValue = InputBox("请输入人数", "计算数开方对话框", 100, , "DEMO. HLP", 10)
```

程序执行到 InputBox 函数语句时，弹出如图 2-3 所示对话框。

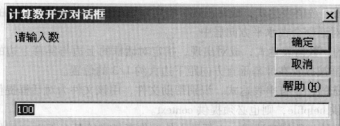


图 2-3 计算数开方对话框

这个函数中有两个可选参数未用所以用两个逗号，另外对话框增加一个帮助键，可以调用帮助信息供用户使用。

注意：这个函数返回值是 String 数据类型，如果需要数值型结果，必须进行类型转换。

例如，还是要求用户输入数，计算该数的开方数。

```
Dim StrMyValue As String
```

```
Dim DblMyValue, x As Double
```

```
StrMyValue = InputBox("请输入数", "计算数开方对话框")
```

```
DblMyValue = Val(StrMyValue)
```

```
x = Sqr(DblMyValue)
```

如果不使用 Val 函数进行类型转换，执行到 Sqr 函数时会提示出错。

2.5.2 输出函数 MsgBox

MsgBox 函数的功能是在对话框中显示消息，等待用户单击按钮，并返回一个整型值告知用户单击了哪一个按钮。

MsgBox 函数显示的图标、消息和命令的按钮个数由函数的参数决定，函数的返回值说明用户选择了哪个按钮。然后程序根据用户选择的按钮用 If 语句决定合适的操作。

函数格式如下：

```
MsgBox (prompt[, buttons][, title][, helpfile, context])
```

参数说明:

prompt: 必需的, 字符串表达式, 作为显示在对话框中的消息。prompt 的最大长度大约为 1024 个字符, 由所用字符的宽度决定。如果 prompt 的内容超过一行, 则可以在每一行之间用回车符(Chr(13))、换行符(Chr(10))或是回车与换行符的组合(Chr(13) & Chr(10))将各行分隔开来。

buttons: 可选的, 数值表达式是值的总和, 指定显示按钮的数目及形式、使用的图标样式、默认按钮是什么, 以及消息框的强制回应等。如果省略, 则 buttons 的默认值为 0。

title: 可选的, 在对话框标题栏中显示的字符串表达式。如果省略 title, 则将应用程序名放在标题栏中。

helpfile: 可选的, 字符串表达式, 识别用来向对话框提供上下文相关帮助的帮助文件。如果提供了 helpfile, 则也必须提供 context。

context: 可选的, 数值表达式, 由帮助文件的作者指定给适当的帮助主题的帮助下文编号。如果提供了 context, 则也必须提供 helpfile。

其中, buttons 的按钮数目和形式是用数值表示的, 有 6 种组合, 每种组合用数字编好码并且规定一个常量的名称, 使用时既可以用常量的名称作参数, 也可以用该组合对应的数字作参数。

输出对话框上按钮的样式见表 2-7。

表 2-7 输出对话框上按钮的样式

数 值	常 量	功 能
0	vbOKOnly	只显示 OK 按钮
1	vbOKCancel	显示 OK 及 Cancel 按钮
2	vbAbortRetryIgnore	显示 Abort、Retry 及 Ignore 按钮
3	vbYesNoCancel	显示 Yes、No 及 Cancel 按钮
4	vbYesNo	显示 Yes 及 No 按钮
5	vbRetryCancel	显示 Retry 及 Cancel 按钮

输出对话框中使用的图标样式, 有 4 种组合, 每种组合用数字编好码并且规定一个常量的名称, 使用时既可以用常量的名称作参数, 也可以用该组合对应的数字作参数。输出对话框上图标样式见表 2-8。

表 2-8 输出对话框上图标的样式

数 值	常 量	功 能
16	vbCritical	显示 Critical Message 图标
32	vbQuestion	显示 Warning Query 图标
48	vbExclamation	显示 Warning Message 图标
64	vbInformation	显示 Information Message 图标

输出对话框中默认按钮有 4 种组合, 每种组合用数字编好码并且规定一个常量的名称, 使用时既可以用常量的名称作参数, 也可以用该组合对应的数字作参数。输出对话框上默认



按钮见表 2-9。

表 2-9 输出对话框上默认按钮

数 值	常 量	功 能
0	vbDefaultButton1	第一个按钮是默认值
256	vbDefaultButton2	第二个按钮是默认值
512	vbDefaultButton3	第三个按钮是默认值
768	vbDefaultButton4	第四个按钮是默认值

输出对话框的强制回应方式有 6 种组合, 每种组合用数字编好码并且规定一个常量的名称, 使用时既可以用常量的名称作参数, 也可以用该组合对应的数字作参数。输出对话框的强制回应见表 2-10。

表 2-10 输出对话框的强制回应

数 值	常 量	功 能
0	vbApplicationModal	应用程序强制返回; 应用程序一直被挂起, 直到用户对消息框作出响应才继续工作
4096	vbSystemModal	系统强制返回; 全部应用程序都被挂起, 直到用户对消息框作出响应才继续工作
16384	vbMsgBoxHelpButton	将 Help 按钮添加到消息框
65536	vbMsgBoxSetForeground	指定消息框窗口作为前景窗口
524288	vbMsgBoxRight	文本为右对齐
1048576	vbMsgBoxRtlReading	指定文本应在希伯来和阿拉伯语系统中的从右到左显示

MsgBox 函数的返回值是整数, 从表中可以看出来, 提供给用户只有 7 种按钮, 把这些按钮编码, 则根据返回值就可以确定用户按动哪个按钮了。MsgBox 函数返回值见表 2-11。

表 2-11 MsgBox 函数返回值

数 值	常 量	功 能	数 值	常 量	功 能
1	vbOK	按下 OK 按钮	5	vbIgnore	按下 Ignore 按钮
2	vbCancel	按下 Cancel 按钮	6	vbYes	按下 Yes 按钮
3	vbAbort	按下 Abort 按钮	7	vbNo	按下 No 按钮
4	vbRetry	按下 Retry 按钮			

可以在程序代码中直接使用这些常数名称, 而不使用实际数值。

例如, 要输出如图 2-4 所示的按钮的数目及形式, 使用的图标样式, 默认按钮是什么以及消息框的强制回应可以用下面语句:

```
Dim Response As Integer
```

```
Response = MsgBox("你想继续操作吗?", vbYesNo + vbQuestion + vbDefaultButton1, "输出对话框",  
Help, Ctxt)
```

也可以:

```
Response = MsgBox("你想继续操作吗?", 4 + 32 + 256, "输出对话框例子")
```

又如:

```
Dim Response As Integer
Response = MsgBox("确实要退出吗?", vbYesNo + vbCritical + vbDefaultButton2, "警告")
```

执行后显示图 2-5 所示的效果。

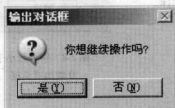


图 2-4 VbQuestion 方式输出对话框

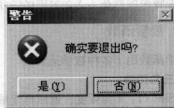


图 2-5 vbCritical 方式输出对话框

注意: 将这些数字相加以生成 buttons 参数值的时候, 只能由每组取值用一个数字, 否则会提示出错。

例如, 使用输入对话框输入一个数, 计算该数的平方根, 如果未输入数据, 提示采用 Information Message 图标; 如果输入的数小于 0 数据, 提示采用 Critical Message 图标。

可进行如下编码:

```
Dim MsgBoxValue As Long
Dim myValue As String
myValue = InputBox("请输入一个整数", "MsgBox 对话框", defaultVal, 100, 100)
If myValue = "" Then
    MsgBoxValue = MsgBox("没有输入任何值!", vbInformation + vbOKOnly, "MsgBox 对话框")
Else
    If Val(myValue) < 0 Then
        MsgBox "输入的值 < 0!", vbExclamation + vbOKOnly, "MsgBox 对话框"
    Else
        MsgBox "输入值是" + myValue + ", 它的平方根是" + Str(Sqr(Val(myValue)))
        + "!", vbInformation + vbOKOnly, "MsgBox 对话框"
    End If
End If
```

正确输入数值, 没有输入任何值和输入负数的显示对话框如图 2-6 ~ 图 2-8 所示。

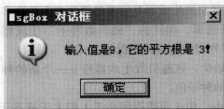


图 2-6 结果输出对话框

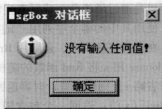


图 2-7 没有输入值输出对话框

2.5.3 类型转换函数

类型转换函数的每个函数都可以强制将一个表达式转换成某种特定数据类型。

类型转换函数在 2.3.5 节中已经介绍, 这里不再详述。

2.5.4 数学函数

数学函数用于各种数学运算。

1. 三角函数

- $\text{Sin}(x)$: x 为弧度, 返回自变量 x 的正弦值。
- $\text{Cos}(x)$: x 为弧度, 返回自变量 x 的余弦值。
- $\text{Tan}(x)$: x 为弧度, 返回自变量 x 的正切值。
- $\text{Atn}(x)$: x 为直角三角形两边的比值, 返回以弧度为单位的角 (x 的反正切值)。

上述各项中 x 为数值表达式, 函数返回值为 `Double` 型。

2. 一般计算

- $\text{Exp}(\text{number})$: 功能是指定 e (自然对数的底) 的某次方, 返回值为 `Double` 型。

注意: 如果 number 的值超过 709.782712893, 则会导致错误发生。常数 e 的值大约是 2.718282。

- $\text{Log}(\text{number})$: 功能是求以 e 为底的对数, 返回值为 `Double` 型。对任意底 n 来计算数值 x 的对数值, 可以将 x 的自然对数值除以 n 的自然对数值。例如, 下面编写一个函数来求以 10 为底的对数值:

```
Static Function Log10(X)
    Log10 = Log(X)/Log(10#)
End Function
```

Exp 函数的作用和 Log 的作用互补, 所以有时也称做反对数。

- $\text{Sqr}(x)$: 功能是求 x 的平方根, 返回值为 `Double` 型。

```
Dim MySqr
MySqr = Sqr(4) '返回 2
```

3. 产生随机数函数

- $\text{Randomize}[x]$: 功能是初始化随机数生成器。

计算机是通过随机数生成器来生成随机数的, 随机数生成器是用某些算法产生的许多数的序列, 使用其中的哪个序列用“随机数种子”来选择, 相同的“随机数种子”生成相同的序列。在选择好序列后, 还要用 Rnd 函数决定序列当中的具体随机数。

Randomize 用 x 将 Rnd 函数的随机数生成器初始化, 该随机数生成器给 x 一个新的种子值。如果省略 x , 则用系统计时器返回的值作为新的种子值。

- $\text{Rnd}[(x)]$: 返回一个包含随机数值的 `Single`。 Rnd 函数返回小于 1 但大于或等于 0 的值。

如果 $x > 0$, 默认值, 以上一个随机数作种子, 产生当前序列中的下一个随机数。

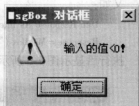


图 2-8 输入数小于 0 输出对话框

如果 $x < 0$ ，以 x 作种子，产生下一个随机数。如果每次都使用 x 作为随机数种子得到的相同结果。

如果 $x = 0$ ，产生与最近生成的随机数相同的数。

如果 x 省略，则在随机数生成器的当前序列中产生下一个随机数。

因每一次连续调用 Rnd 函数时都用序列中的前一个数作为下一个数的种子，所以对于任何最初给定的种子都会生成相同的数列。

在调用 Rnd 之前，先使用无参数的 Randomize 语句初始化随机数生成器，该生成器具有基于系统计时器的种子。

注意：要产生指定范围的随机整数，可用以下公式： $\text{Int}((\text{upperbound} - \text{lowerbound} + 1) * \text{Rnd} + \text{lowerbound})$ ，其中，upperbound 是此范围的上界，而 lowerbound 是此范围内的下界。

例如，要产生一个 20 与 69 之间的随机数，只要 $\text{Int}((69 - 20 + 1) * \text{Rnd} + 20)$ 即可。

4. 取得绝对值函数

■ Abs(x)：功能是返回自变量 x 的绝对值。

例如，ABS(-1) 和 ABS(1) 都返回 1。

5. 取得表达式的正负号函数

■ Sgn(x)：返回自变量 x 的符号，即当 x 为负数时，返回 -1；当 x 为 0 时，返回 0；当 x 为正数时，返回 1。

例如：

MySign = Sgn(9)，则 MySign = 1

MySign = Sgn(0)，则 MySign = 0

MySign = Sgn(-9)，则 MySign = -1

6. 数值变换函数

■ Fix(x)：返回函数的整数部分，功能是去掉一个浮点数的小数部分，保留其整数部分。

■ Int(x)：返回函数的整数部分，功能是求不大于自变量 x 的最大整数。

Int 和 Fix 都会删除 number 的小数部分而返回剩下的整数。Int 和 Fix 的不同之处在于，如果 number 为负数，则 Int 返回小于或等于 number 的第一个负整数，而 Fix 则会返回大于或等于 number 的第一个负整数。

例如，Int(-8.3) 返回值是 -9，而 Fix(-8.3) 返回值是 -8。Int(8.3) 返回值是 8，而 Fix(8.3) 返回值是 8。

■ Round(expression[, numdecimalplaces])：返回一个数值，该数值是按照指定的小数位数进行四舍五入运算的结果。其中，expression 是必需的，要进行四舍五入运算的数值表达式；numdecimalplaces 可选的，数字值，表示进行四舍五入运算时，小数点右边应保留的位数。如果忽略，则 Round 函数返回整数。

例如：

Round(8.3) 返回值是 8，而 Round(8.6) 返回值是 9。

Round(8.321, 2) 返回值是 8.32，Round(-8.321, 2) 返回值是 -8.32。

Round(-8.326, 2) 返回值是 -8.33，Round(8.326, 2) 返回值是 8.33。

2.5.5 日期与时间函数

1. 设置或返回当前日期或时间

■ **Date**: 设置或返回包含系统当前日期。

Dim MyDate

MyDate = Date

将 MyDate 的值设置为系统当前的日期,也可以设置当前系统时间。

例如:

Date = #2/3/1999# '将系统当前的日期变为 1999 年 2 月 3 日

■ **Time**: 设置或返回包含系统当前时间。

Time 用法与 Date 类似。

例如:

Time = #6:30:00 AM# '将系统当前的时间变为上午 6 点 30 分

■ **Now**: 返回包含系统当前日期和时间。

例如:

Dim Today

Today = Now '将系统当前的日期与时间给变量

2. 返回当前年、月、日或星期

■ **Day(Now)**: 返回当前的日期,其值为 1 到 31 之间的整数。

例如:

Dim MyDate, MyDay

MyDate = #February 22, 1999# '指定一日期

MyDay = Day(MyDate) 'MyDay 的值为 22

■ **Month(Now)**: 返回当前的月份,其值为 1 ~ 12 之间的整数。

例如:

Dim MyDate, MyMonth

MyDate = #February 22, 1999# '指定一日期

MyMonth = Month(MyDate) 'MyMonth 的值为 22

■ **Year(Now)**: 返回当前的年份。

例如:

Dim MyDate, MyYear

MyDate = #February 22, 1999# '指定一日期

MyYear = Year(MyDate) 'MyYear 的值为 1999

■ **WeekDay(Now)**: 返回当前的星期。

例如:

Dim MyDate, MyWeekDay

MyDate = #February 22, 1999# '指定一日期

MyWeekDay = Weekday(MyDate) 'MyWeekDay 的值为 1, 因为 MyDate 是星期一

3. 返回当前小时、分、秒

■ **Hour(Now)**: 其值为 0 ~ 23 之间的整数,表示一天之中的某一钟点。

- Minute(Now): 其值为 0 ~ 59 之间的整数, 表示一小时中的某一分钟。
- Second(Now): 其值为 0 ~ 59 之间的整数, 表示一分钟之中的某一秒。

例如:

```
Dim MyTime, MyHour, MyMinute, MySecond
MyTime = #6:15:27 PM# '指定一时间
MyHour = Hour(MyTime) 'MyHour 的值为 18
MyMinute = Minute(MyTime) 'MyMinute 的值为 15
MySecond = Second(MyTime) 'MySecond 的值为 27
```

4. 其他日期与时间函数

- DateAdd: 已知一个日期还加上了一段时间间隔, 返回得到的新日期。

函数格式如下:

DateAdd(interval, number, date)

参数说明:

- 1) date 是必要参数, 已知的日期。
- 2) number 是必要参数, 数值表达式, 是要加上的时间间隔的数目。其数值可以为正数(得到未来的日期), 也可以为负数(得到过去的日期)。
- 3) interval 是必要参数, 字符串表达式, 是所要加上去的时间间隔的类型。interval 参数值见表 2-12。

表 2-12 interval 参数值

interval 值	含 义	interval 值	含 义
"Yyyy"	年	"w"	一周的日数
"q"	季	"Ww"	周
"M"	月	"H"	时
"Y"	一年的日数	"N"	分钟
"D"	日	"S"	秒

例如:

```
Dim FirstDate As Date '声明变量
Dim Number As Integer
Dim Msg
FirstDate = InputBox("请输入一个日期")
Number = InputBox("请输入要加的天数")
Msg = "New date: " & DateAdd("D", Number, FirstDate)
MsgBox Msg
```

注意: 因为一年中的各月的天数不同, 所以一个日期加一个月和加 30 天的值是不同的。

例如:

```
Dim MyString
```

```
MyString = DateAdd("D",30,#1/31/1996#)
```

```
MsgBox(MyString)
```

程序运行后弹出如图 2-9 所示对话框。

```
MyString = DateAdd("M",1,#1/31/1996#)
```

```
MsgBox(MyString)
```

程序运行后弹出如图 2-10 所示对话框。



图 2-9 结果输出对话框 1



图 2-10 结果输出对话框 2

■ **DateDiff**: 返回两个指定日期之间的时间间隔数目。

函数格式如下:

```
DateDiff(interval,date1,date2[,firstdayofweek[,firstweekofyear]])
```

参数说明:

- 1) date1 和 date2 是必要参数, 已知的日期。计算中要用到的两个日期。
- 2) interval 是必要参数, 字符串表达式, 表示用来计算 date1 和 date2 的时间差的时间间隔。同 DateAdd 函数的 interval 参数值列表。
- 3) firstdayofweek 是可选参数, 会影响使用时间间隔符号 "W" 或 "WW" 计算的结果。firstweekofyear 是可选参数, 会影响使用时间间隔符号 "Y" 或 "Yyyy" 计算的结果。

例如, DateDiff 函数显示今天与给定日期之间间隔天数:

```
DiffDate = "从当天开始的天数:" & DateDiff("D",Now,theDate)
```

2.5.6 字符串函数

1. 字符串截取函数

字符串截取函数用来截取字符串的一部分, 可以从字符串的左部、右部或中部截取。

■ **Left** 函数: 功能是返回“字符串”的前若干个字符。

函数格式如下:

```
Left(string,length)
```

参数说明:

两个参数都是必需的。其中的“字符串”可以是字符串常量、字符串变量、字符串函数或字符串连接表达式。length 指出将返回多少个字符, 如果为 0, 返回零长度字符串(""), 如果大于或等于 string 的字符数, 则返回整个字符串。

例如:

```
Dim AnyString, MyStr
AnyString = "abcdefg" '定义字符串
MyStr = Left(AnyString,1) '返回"a"
MyStr = Left(AnyString,3) '返回"abc"
```

MyStr = Left(AnyString, 20) '返回"abcdefg"

■ Mid 函数: 功能是从第 start 个字符开始, 向后截取 length 个字符。
函数格式如下:

Mid(string, start[, length])

参数说明:

string 和 start 两个参数都是必需的。Length 是可选参数, 如果省略或 length 超过文本的字符数(包括 start 处的字符), 将返回字符串中从 start 到尾端的所有字符。

例如:

```
Dim MyString, FirstWord, LastWord, MidWords
```

```
MyString = "Mid Function Demo" '建立一个字符串
```

```
FirstWord = Mid( MyString, 1, 3) '返回"Mid"
```

```
LastWord = Mid( MyString, 14, 4) '返回"Demo"
```

```
MidWords = Mid( MyString, 5) '返回"Function Demo"
```

■ Right 函数: 功能是返回“字符串”的最后若干个字符。

函数格式如下:

Right(string, length)

参数说明:

两个参数都是必需的。length 指出将返回多少个字符, 如果为 0, 返回零长度字符串(""), 如果大于或等于 string 的字符数, 则返回整个字符串。

例如:

```
Dim AnyString, MyStr
```

```
AnyString = "Hello World" '定义字符串
```

```
MyStr = Right( AnyString, 1) '返回"d"
```

```
MyStr = Right( AnyString, 6) '返回"World"
```

```
MyStr = Right( AnyString, 20) '返回"Hello World"
```

2. 删除空白字符函数

■ Ltrim: 将某字符串的开头的空格全部去除。

■ Rtrim: 将某字符串的结尾的空格全部去除。

■ Trim: 将某字符串的开头及结尾的空格全部去除。

函数格式如下:

Ltrim(string)

Rtrim(string)

Trim(string)

参数说明:

string 参数是必需的, 可以是任何有效的字符串表达式。如果 string 包含 Null, 将返回 Null。

空白字符包括空格键、Tab 键等。

例如:

```
Ltrim(" Good Mornig")的返回值是"Good Mornig"
```

```
Rtrim(" Good Mornig")的返回值是"Good Mornig"
```

Trim(" Good Mornig ")的返回值是"Good Mornig"

3. 计算字符串长度

■ Len 函数: 返回字符串的长度, 或是存储一变量所需的字节数。
函数格式如下:

Len (string|varname)

参数说明:

- 1) 两个可能的参数必须有其一(而且只能有其一)。
- 2) 如果参数为 string, Len 函数返回字符串的长度, string 可以是任何有效的字符串表达式。当 string 包含 Null, 会返回 Null。
- 3) 如果参数为 varname, Len 函数返回存储一变量所需的字节数。varname 可以是任何有效的变量名称。如果 varname 包含 Null, 会返回 Null。

例如:

```
Dim MyInt As Integer, MyCur As Currency
Dim MyString, MyLen
MyString = "Hello World" '设置变量初值
MyLen = Len(MyInt) '返回 2
MyLen = Len(MyString) '返回 11
MyLen = Len(MyCur) '返回 8
```

4. 建立重复字符的字符串

■ Space 函数: 功能是返回特定数目空格的。

函数格式如下:

Space (number)

参数说明:

number 参数为必要的, 为字符串中想要的空格数。

例如:

```
Dim MyString
MyString = Space(6) '返回 6 个空格的字符串
MyString = "Visual" & Space(2) & "Basic" '将 2 个空格插入两个字符串中间
```

■ String 函数: 功能是返回包含指定长度重复字符的字符串。

函数格式如下:

String (number, character)

参数说明:

- 1) number 为必要参数; 返回的字符串长度。如果 number 包含 Null, 将返回 Null。
- 2) character 为必要参数; 为指定字符的字符码或字符串表达式, 其第一个字符将用于建立返回的字符串。如果 character 包含 Null, 就会返回 Null。

例如:

```
Dim MyString
MyString = String(6, "#") '返回"#####"
```

```
MyString = String(3, 42) '返回"***"
```

如果 Character 是的字符码或字符串表达式, 其第一个字符将用于建立返回的字符串。

例如:

```
MyString = String(8, "xyz") '返回"xxxxxxx"
```

5. 字符串匹配函数

■ InStr 函数: 查找某字符串在另一个字符串中首次出现的位置。

函数格式如下:

```
InStr([start,] string1, string2[, compare])
```

参数说明:

- 1) start 为可选参数, 为数值表达式, 设置每次搜索的起点。如果省略, 将从第一个字符的位置开始, 如果 start 包含 Null, 将发生错误, 如果指定了 compare 参数, 则一定要有 start 参数。
- 2) string1 为必要参数, 接受搜索的字符串表达式。
- 3) string2 为必要参数, 被搜索的字符串表达式。
- 4) compare 为可选参数, 指定字符串比较。compare 等于 0 执行一个二进制比较, 这时字符串中大写和小写字母认为是相同的; compare 等于 1, 执行一个按照原文比较, 这时字符串中大写和小写字母认为是不同的。如果省略 compare, 选项 compare 的设置将决定比较的类型。

例如:

```
Dim SearchString, SearchChar, MyPos
```

```
SearchString = "XXpXXpXXpXXp" '被搜索的字符串
```

```
SearchChar = "P" '要查找字符串"P"
```

```
'从第四个字符开始, 以文本比较的方式找起。返回值为 6(小写 p)
```

```
'小写 p 和 大写 P 在文本比较下是一样的
```

```
MyPos = InStr(4, SearchString, SearchChar, 1)
```

```
'从第一个字符开始, 以二进制比较的方式找起。返回值为 9(大写 P)
```

```
'小写 p 和 大写 P 在二进制比较下是不一样的
```

```
MyPos = InStr(1, SearchString, SearchChar, 0)
```

```
'默认的比对方式为二进制比较(最后一个参数可省略)
```

```
MyPos = InStr(SearchString, SearchChar) '返回 9
```

```
MyPos = InStr(1, SearchString, "W") '返回 0
```

6. 大小写变换函数

■ LCase 函数: 将字符串转成全部小写。

函数格式如下:

```
LCase(string)
```

参数说明:

string 参数是必要的, 可以是任何有效的字符串表达式。如果 string 包含 Null, 将返回 Null。

注意: LCase 函数只使大写的字母会转成小写; 所有小写字母和非字母字符保持不变。

例如:

```
Dim UpperCase, LowerCase
```

```
UpperCase = "Hello" '要输送的字符串
```

```
LowerCase = LCase(UpperCase) '返回"hello"
```

■ UCase(string): 将字符串转成全部大写。

函数格式如下:

UCase(string)

参数说明:

string 参数是必要的, 可以是任何有效的字符串表达式。如果 string 包含 Null, 将返回 Null。

同样, Ucase 函数只使小写的字母转成大写; 原本大写或非字母之字符保持不变。

例如:

```
Dim LowerCase, UpperCase
```

```
LowerCase = "Hello"
```

'要输送的字符串

```
UpperCase = UCase(LowerCase)
```

'返回"Hello"

7. 比较两个字符串

■ StrComp 函数: 用来比较两个字符串。如果第三个参数值为 1, 字符串是以文本比较的方式进行比较; 如果第三个参数值为 0 或是默认, 则以二进制比较的方式进行比较。

函数格式如下:

StrComp(string1, string2[, compare])

参数说明:

1) string1 为必要参数, 可以是任何有效的字符串表达式。

2) string2 为必要参数, 可以是任何有效的字符串表达式。

3) compare 为可选参数。指定字符串比较的类型。其中, compare 等于 0, 执行一个二进制比较, compare 等于 1, 执行一个按照文本的比较。文本比较方式会将大小写字母视为一样, 但二进制比较方式则视为不同。compare 如果默认, 默认值为 0。

4) StrComp 的返回值: 如果 string1 小于 string2 为 -1; 如果 string1 等于 string2 为 0; 如果 string1 大于 string2 返回值为 1。

例如:

```
Dim MyStr1, MyStr2, MyComp
```

```
MyStr1 = "XYZ"
```

```
MyStr2 = "xyz"
```

```
MyComp = StrComp(MyStr1, MyStr2, 1)
```

'返回 0

```
MyComp = StrComp(MyStr1, MyStr2, 0)
```

'返回 -1

```
MyComp = StrComp(MyStr2, MyStr1)
```

'返回 1

8. 求 ASCII 与 ANSI 值

■ Asc 函数: 返回一个 Integer, 代表字符串中首字母的字符代码。

函数格式如下:

Asc(string)

参数说明:

string 为必要参数, 可以是任何有效的字符串表达式。

例如:

```
Dim MyNumber
```

```
MyNumber = Asc("A") '返回 65
```

```
MyNumber = Asc("a") '返回 97
```

```
MyNumber = Asc("Apple") '返回 65
```

■ Chr 函数：返回指定字符码所代表的字符。

函数格式如下：

```
Chr(charcode)
```

参数说明：

charcode 参数是必要的，可以是一个用来识别某字符的 Long。

例如：

```
Dim MyChar
```

```
MyChar = Chr(65) '返回 A
```

```
MyChar = Chr(97) '返回 a
```

此外，还有设置字符串格式的 Format，将在第 3 章中介绍。

2.5.7 目录和文件函数

■ CurDir 函数：返回当前的路径。

函数格式如下：

```
CurDir[(drive)]
```

参数说明：

drive 参数是可选的，是一个字符串表达式，它指定一个存在的驱动器。如果没有指定驱动器，或 drive 是零长度字符串("")，则 CurDir 会返回当前驱动器的路径。

例如：

```
'假设 C 驱动器的当前路径为"c:\Program Files"
```

```
'假设 D 驱动器的当前路径为"d:\abc"
```

```
'假设 C 为当前的驱动器
```

```
Dim MyPath
```

```
MyPath = CurDir '返回"c:\Program Files"
```

```
MyPath = CurDir("C") '返回"c:\Program Files"
```

```
MyPath = CurDir("D") '返回"d:\abc"
```

■ Dir 函数：返回一个 String，用以表示一个文件名、目录名或文件夹名称。

函数格式如下：

```
Dir[(pathname[,attributes])]
```

参数说明：

1) pathname 为可选参数。用来指定文件名的字符串表达式，可能包含目录或文件夹及驱动器。如果没有找到 pathname，则会返回零长度字符串("")。pathname 参数见表 2-13。

2) attributes 为可选参数。常数或数值表达式，其总和用来指定文件属性。如果省略，则会返回匹配 pathname 但不包含属性的文件。



表 2-13 pathname 参数

常 量	值	含 义
vbNormal	0	(默认)指定没有属性的文件
vbReadOnly	1	指定无属性的只读文件
vbHidden	2	指定无属性的隐藏文件
vbSystem	4	指定无属性的系统文件
vbVolume	8	指定卷标文件; 如果指定了其他属性, 则忽略 vbVolume
vbDirectory	16	指定无属性文件及其路径和文件夹

例如:

'返回 abc.txt (如果该文件存在)

MyFile = Dir("c:\Program Files\abc.txt")

另外, Dir 支持多字符(*)和单字符(?)的通配符来指定多重文件。

例如:

'返回带指定扩展名的文件名。如果超过一个 "*.ini" 文件存在

'函数将返回按条件第一个找到的文件名

MyFile = Dir("c:\Windows*.ini")

■ GetAttr 函数: 返回一个文件、目录、或文件夹的属性。

函数格式如下:

GetAttr(pathname)

参数说明:

pathname 是必要的参数, 用来指定一个文件名、目录或文件夹、以及驱动器。

GetAttr 返回的值, 是下面这些属性值的总和, 见表 2-14。

表 2-14 GetAttr 返回的值含义表

常 数	值	含 义	常 数	值	含 义
vbNormal	0	常规	vbDirectory	16	目录或文件夹
vbReadOnly	1	只读	vbArchive	32	上次备份以后, 文件已经改变
vbHidden	2	隐藏	vbAlias	64	指定的文件名是别名
vbSystem	4	系统文件			

例如:

Dim MyAttr

'假设 TESTFILE 具有隐含属性

MyAttr = GetAttr("TESTFILE") '返回 2

2.5.8 数组函数

■ UBound 函数: 返回数组的指定维数的最大可用下标。

函数格式如下:

UBound(arrayname[, dimension])

■ **LBound 函数**：返回数组的指定维数的最小可用下标。

函数格式如下：

LBound(arrayname[, dimension])

参数说明：

Arrayname 是必需的，数组变量的名称，遵循标准的变量命名约定。dimension 是可选的；指定返回哪一维的下界。1 表示第一维，2 表示第二维，如此类推。如果省略 dimension，就认为是 1。

例如：

```
Dim Lower, Upper
```

```
Dim MyArray(1 To 10, 5 To 15, 10 To 20) '声明数组变量
```

```
Lower = LBound(MyArray, 1) '返回 1
```

```
Lower = LBound(MyArray, 3) '返回 10
```

```
Upper = UBound(MyArray, 1) '返回 10
```

```
Upper = UBound(MyArray, 3) '返回 20
```

```
Upper = UBound(AnyArray) '返回 10
```

■ **Array 函数**：返回一个的数组。

函数格式如下：

Array(arglist)

参数说明：

arglist 是一个用逗号隔开的值表。

例如：

```
Dim MyWeek, MyDay
```

```
MyWeek = Array("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")
```

```
'返回值假设下界的设置为 1 (使用 Option Base 语句)
```

```
MyDay = MyWeek(2) 'MyDay 的值为 "Tue"
```

```
MyDay = MyWeek(4) 'MyDay 的值为 "Thu"
```

2.5.9 其他常用函数

■ **Hex 函数**：一个十进制数转换为十六进制数。

■ **Oct 函数**：把一个十进制数转换为八进制数。

函数格式如下：

Hex(number)

Oct(number)

参数说明：

number 是必要的参数，为任何有效的数值表达式或字符串表达式。

例如：

```
Dim MyOct, MyHex
```

```
MyOct = Oct(4) '返回 4
```

```
MyOct = Oct(8) '返回 10
```

```
MyHex = Hex(5) '返回 5
```

MyHex = Hex(10) '返回 A

■ RGB 函数：用来表示一个 RGB 颜色值。

函数格式如下：

RGB(red, green, blue)

例如：

```
Dim RED, I, RGBValue, MyObject
```

```
Red = RGB(255, 0, 0) '返回代表红色的值
```

```
I = 75 '初始化偏移量
```

```
RGBValue = RGB(I, 64 + I, 128 + I) '同 RGB(75, 139, 203)
```

```
MyObject.Color = RGB(255, 0, 0) '设置 MyObject 的 Color 属性为红色
```

■ Shell 函数：执行一个可执行文件，返回一个 Variant(Double)，如果成功的话，代表这个程序的任务 ID，若不成功，则会返回 0。

函数格式如下：

Shell(pathname[, windowstyle])

参数说明：

pathname 为必需参数，类型为 String，它指出了要执行的程序名，以及任何需要的参数或命令行变量，也可以包括路径名。windowstyle 为可选参数，Integer 类型，指定在程序运行时窗口的样式。windowstyle 的值：VbHide, 0，窗口被隐藏，且焦点会移到隐式窗口；VbNormalFocus, 1，窗口具有焦点，且会还原到它原来的大小和位置；VbMinimizedFocus, 2，窗口会以一个具有焦点的图标来显示（默认值）；VbMaximizedFocus, 3，窗口是一个具有焦点的最大化窗口；VbNormalNoFocus, 4，窗口会被还原到最近使用的大小和位置，而当前活动的窗口仍然保持活动；VbMinimizedNoFocus, 6，窗口会以一个图标来显示，而当前活动的窗口仍然保持活动。

例如：

```
'将第二个参数值设成 1，可让该程序以正常大小的窗口完成，并且拥有焦点
```

```
Dim RetVal
```

```
RetVal = Shell("C:\WINDOWS\CALC.EXE", 1) '完成 Calculator
```

还有分支函数在第 4 章介绍。

2.6 Visual Basic 程序的注释和书写规范

2.6.1 Visual Basic 程序的注释

1. 什么是注释

用来解释代码如何工作的附加文本。在 VB 中，注释以单引号(')开头，或者在空格之后跟 Rem 关键字，再在其后写注释。

2. 程序的注释方式

VB 程序的注释方式有 3 种：

1) 整行注释一般以 Rem 开头，也可以用单引号(')，以 Rem 开头要求注释在语句之后

要用冒号隔开。

2) 用单引号(')引导的注释,既可以是整行的,也可以直接放在语句的后面,最方便。

3) 利用“编辑”工具栏的“设置注释块”、“解除注释块”可以同时设置多行注释。这种方法设置的注释使用的是单引号(')。

例如:

```
MyStr1 = "Hello" : Rem 注释在语句之后要用冒号隔开
```

```
MyStr2 = "Goodbye"
```

```
'这也是一条注释,无需使用冒号
```

2.6.2 Visual Basic 程序的书写规范

1. 语句构成

语句是 VB 程序的最基本成分。语句的一般形式如下:

[语句定义符] <语句体>

其中,语句定义符用于规定语句功能,此部分一般可省略;语句体用于向系统提供某些必要的说明内容或者规定系统应执行的某些操作。

2. 代码书写规则

1) VB 程序中允许将单行语句分成多行书写;方法是用续行符(一个空格后面跟一个下划线)表明本条语句未完,接下一行,从而将长语句分成多行。

例如:

```
Data1.RecordSource = _
```

```
"SELECT * FROM Titles,Publishers _
```

```
& "WHERE Publishers.PubId = Titles.PubID _
```

```
& "AND Publishers.State = 'CA'"
```

2) VB 程序中也允许将多个语句合并到同一行上,方法是用冒号(:)将它们分开。

例如:

```
a = b; b = c; c = a
```

3) 一行允许多达 255 个字符。

4) 程序中不区分字母的大小写,Ab 与 AB 等效。

5) 系统对用户程序代码进行自动转换:

■ 对于 VB 中的关键字,首字母被转换成大写,其余转换成小写。

■ 若关键字由多个英文单词组成,则将每个单词的首字母转换成大写。

■ 对于用户定义的变量、过程名,以第一次定义的为准,以后输入的自动转换成首次定义的形式。

3. 数值表示

VB 中使用的数值通常用十进制表示的。也可以用前缀 &H 表示十六进制数,而用 &O 表示八进制数。无表示的是十进制数。

例如,对十进制数字 9,在 VB 中可以表示成 9、&O11 或 &H9。

4. VB 的命名约定

在编写 VB 代码时,要声明和命名许多元素(Sub 和 Function 过程、变量、常数等)。这些命名标识符。

在 VB 代码中命名标识符时, 必须遵循这些规则:

- 它们必须以字母开头。
- 它们不可以包含嵌入的句号或者类型声明字符(规定数据类型的特殊字符)。
- 它们不能超过 255 个字符。控件、窗体、类和模块的名字不能超过 40 个字符。
- 它们不能和受到限制的关键字同名。

受到限制的关键字是 VB 使用的词, 是语言的组成部分。其中包括预定义语句(比如 If 和 Loop)、函数(比如 Len 和 Abs)和操作符(比如 Or 和 Mod)。

习 题

2-1 VB 中有哪几种数据类型?

2-2 如何使用常量?

2-3 如何使用变量?

2-4 变量根据作用域分为过程级变量、模块级变量、全局变量 3 种, 它们的区别是什么?

2-5 在 VB 中, 对于没有赋值的变量, 系统默认值是什么?

2-6 下列哪些符号不能作为 VB 的标识符?

(1) XYZ (2) True1 (3) False (4) 1ABC (5) A[7]

(6) Y_1 (7) IntA (8) A-2 (9) A.3 (10) "Comp"

2-7 下列数据哪些是变量? 哪些是常量? 是什么类型的常量?

(1) name (2) "name" (3) False (4) if (5) "11/16/99"

(6) ej (7) "120" (8) n (9) #11/16/2000# (10) 12.345

2-8 在 VB 中, 下列运算符优先级最高的是:

A. * B. \ C. < D. Not

2-9 用函数获取一个指定目录下的所有文件名。

2-10 表达式 $2 + 5 \setminus 6 * 7 / 8 \text{Mod} 9$ 的值是:

A. 2 B. 3 C. 4 D. 5

2-11 在 VB 中, 声明全局变量所使用的关键字是:

A. Dim B. Public C. Static D. Auto

2-12 数学中 $\sin 57^\circ$ 在 VB 中, 应怎样表示?

2-13 Instr(3, "A12a34A56", "A"), Instr(3, "A12a34A56", "A", 1), Instr("A12a34A56", "A") 的结果分别是多少?

2-14 使用 Dim 和 Static 定义变量有何区别?

2-15 编写一段程序产生 10 个 0~100 之间的随机整数。

2-16 编写一个程序, 从字符串 "Visual Basic" 中找出某个指定字符空格, 以此字符为界拆分成两个字符串。

2-17 使用 MsgBox 函数显示图 2-11 所示输出对话框。

2-18 使用 InputBox 函数显示图 2-12 所示输入对话框。

2-19 编写一个程序, 用鼠标单击窗体时, 弹出图 2-12 所示的输入对话框, 输入姓名, 单击“确定”按钮, 显示图 2-13 所示输出对话框。单击“取消”按钮, 返回主窗体。

2-20 编写一段程序计算现在距离 2008 年 8 月 8 日奥运会开幕还有多少天。

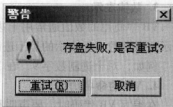


图 2-11 习题 2-17 图

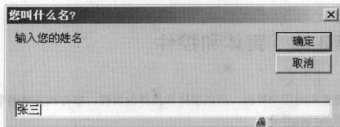


图 2-12 习题 2-18 图

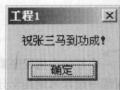


图 2-13 习题 2-19 图

2-21 在程序中使用注释的作用是什么? 如何进行注释?

2-22 简述 VB 程序的书写规范。

第3章 窗体和控件

一个好的应用程序要有美观实用的用户界面，所以窗体和控件的属性、事件和方法的学习对程序设计来说是十分重要和基础的部分。

3.1 赋值语句和赋值相容

3.1.1 赋值语句

赋值语句是为变量或属性赋值的语句。

语句格式如下：

变量名 = 表达式

其中，“=”称为赋值号。

赋值语句作用有两个：

- 1) 计算：将赋值号右边表达式的值计算出来。
- 2) 赋值(存储)：将计算出来表达式的值赋给左边的变量，即存入变量所代表的内存单元。

例如， $x = x * 3$ 功能是将变量 x 的值乘以 3 后又赋值给 x 。

在 VB 中经常可以用到如下几种赋值语句：

- 1) 直接把数值赋给变量：

```
Dim x As Integer
Dim xStr As String
x = 12
xStr = "abcdefg"
```

- 2) 将一个变量的值赋给另一个变量：

```
a = 21
b = a      '这时 a、b、c 中都是 21
```

- 3) 将一个表达式的值赋给一个变量：

```
Dim x As Integer, y As Integer, z As Integer
x = 16
y = 12
z = x * 2 + y * 3
```

- 4) 用赋值语句给对象的某属性设置属性值(有的属性只能通过代码赋值)，它的一般格式如下：

对象名. 属性 = 属性值

例如：

```
Form1.caption = "我的 VB 程序"
```

3.1.2 赋值相容

赋值语句左侧的标识符无论是变量还是控件属性，都代表一个存储单元，当执行赋值语句时，如果这个存储单元的结构和大小能够正确地表示赋值号右边表达式的值，则该赋值语句的值是相容的。

如果表达式值的类型与变量(或控件属性)的类型相同，它们当然赋值相容。如果赋值号左右两边的数据类型不相同，也是需要相容的，只有赋值相容的数据之间才能进行运算操作，否则就会出现“类型不匹配”等错误。VB 对赋值相容的数据类型之间的运算和赋值提供了数据类型自动转换机制。

比较常见的情况有两种：

(1) 数值型与数值型的字符串型的转换

例如：

```
Dim intX As Integer
Dim strY As String
strY = "123.9"
intX = strY
strX = 987
strY = strX
```

上面程序能够正确运行。

又如：

```
Dim intX As Integer
Dim strZ As String
strZ = "hello"
intX = strZ
```

而这个程序运行后会提示“类型不匹配”错误，因为系统对非数值型的字符串不能够自动转换成数值类型。

(2) Numeric 数据类型之间的转换

系统提供的 Numeric 数据类型有几种，这些种数据类型之间是赋值相容的，相互赋值后不会出错。例如：

```
Dim intX As Integer
Dim lngY As Long
intX = 1234567890
lngY = intX
intX = lngY
```

注意：不同的 Numeric 数据类型之间的赋值后，结果的精度将可能会受到影响。

3.1.3 Set 语句

Set 语句功能是将对象引用赋给变量或属性。

语句格式如下：

Set 对象名 = [[New]对象表达式] Nothing

使用说明：

对象名是必需的，可以是变量或属性的名称，遵循标准变量命名约定。

New 是可选的。通常在声明时使用 New，以便可以隐式创建对象。如果 New 与 Set 一起使用，则将创建该类的一个新实例。如果对象名包含了一个对象引用，则在赋值时释放该引用。

对象表达式必需的。由对象名、所声明的相同对象类型的其他变量，或者返回相同对象类型的函数或方法所组成的表达式。

Nothing 是可选的，断绝该对象名与任何指定对象的关联。若没有其他变量指向该对象名原来所引用的对象，将其赋为 Nothing 会释放该对象所关联的所有系统及内存资源。

注意：为对象引用赋值必须使用 Set 语句。

3.2 输入/输出操作

3.2.1 输入操作

绝大多数的应用程序都有需要用键盘输入信息的情况，否则，应用程序不允许用户输入数据，程序只具有浏览功能，很不灵活。因此，输入语句的功能十分必要。

在 VB 系统中，可以通过系统提供的 InputBox 函数获得输入数据，或者通过用户界面上的文本框的内容、命令按钮、复选框、单选按钮、列表框、组合框等控件获得输入数据。

例如：

```
x = text1.Text
y = InputBox("请输入姓名：")
```

3.2.2 输出操作

同输入操作一样，输出也是应用程序不可缺少的部分。如果没有输出，用户根本不知道程序完成了什么功能和结果。

在 VB 系统中，也可以有几种方法进行输出操作：

1. 通过系统提供的 MsgBox 函数输出数据

例如：

```
MsgBox("你的密码有错，请重新输入！")
```

2. 通过用户界面上的文本框等可视控件的属性获得输出结果

例如：

```
Text1.Text = x
```

3. 调用 Print 方法

格式如下：

对象名.Print 输出项列表

使用说明：

(1) 具有 Print 方法的对象

窗体、立即窗口、图片框、打印机等对象都具有 Print 方法。如果省略对象名,则在当前窗体上输出。例如:

如果要在窗体 Form1 上显示"abc",则

```
Form1.Print "abc"
```

或者

```
Print "abc"
```

如果要在立即窗口上显示"abc",则

```
Debug.Print "abc"
```

如果要在打印机上打印字符串"abc",则

```
Printer.Print "abc"
```

(2) 输出列表的格式

格式如下:

{Spc(n)|Tab(n)} 表达式 间隔符号

使用说明:

Spc(n) 是可选的,用来在输出中插入空白字符,具体功能下节介绍。

Tab(n) 是可选的,用来将插入点定位在绝对列号上,具体功能下节介绍。

表达式是可选的,为打印的常数、数值表达式或字符串表达式。表达式如果是常数参数直接打印;如果是数值表达式或字符串表达式则打印表达式的计算结果。

间隔符号指定下个字元的插入点,使用分号(;)的话,插入点会接在上个字元所显示地方的后面。参数 n,使用 Tab(n) 的话,会移动插入点到某行。如果没有该间隔符号的 Tab(n) 或逗号(,),会移动插入点到下一个显示区的开头位置。如果省略间隔符号,下一字元将会出现在下一列。

例如:

```
Print 6
```

```
Print "abc"
```

```
Print 2 + 6
```

```
Print "a"+"b"+"c"
```

当前窗体上输出的结果如下:

```
6
```

```
abc
```

```
8
```

```
abc
```

(3) 多个表达式的分隔

如果有多个表达式同时输出,则表达式之间用逗号、分号和空格来分隔。

使用逗号时,下一个输出项内容显示在下一个标准打印区(输出每行 14 个字符为一个标准打印区);使用分号和空格时,下一个输出项内容直接显示在上一个输出项的后面。



例如:

```
Debug.Print "abc";"efg"
```

立即窗口上输出的结果如下:

```
abcefg
```

又如:

```
Debug.Print "abc","efg"
```

立即窗口上输出的结果如下:

```
abc    efg
```

如果一条 Print 语句的最后一个输出项后面有逗号或分号,则下一次执行 Print 语句时,输出的内容不换行;否则,换行显示。

例如:

```
Print "1+2=";
```

```
Print 1+2
```

当前窗体上输出的结果如下:

```
1+2=3
```

3.2.3 常用的输出格式控制函数和方法

为了控制输出格式,常使用如下几个输出格式控制函数。

1. Tab 函数

格式如下:

```
Tab[(n)]
```

功能:与 Print 语句一起使用,对输出进行绝对地址定位。

使用说明:

参数 n 是可选的,在显示或打印列表中的下一个表达式之前移动的列数。若省略此参数,则 Tab 将插入点移动到下一个打印区的起点。

例如:

```
Debug.Print Tab(10);"10 columns from start."
```

则在立即窗口的第 10 列显示 "10 columns from start." 字符串。

如果在当前窗体的输出学生成绩,格式为

学号 姓名 成绩

2007001	张三	87
2007002	李四	89
2007003	王五	96

则应如下编码:

```
Print Tab(15);"学号";Tab(26);"姓名";Tab(36);"成绩"
```

```
Print Tab(14);String(27,"-") 输出 27 个减号字符"-"
```

```
Print Tab(14);"2007001";Tab(26);"张三";Tab(36);87
```

```
Print Tab(14);"2007002";Tab(26);"李四";Tab(36);89
Print Tab(14);"2007003";Tab(26);"王五";Tab(36);96
Print Tab(14);String(27,"-")    '输出 27 个减号字符"-"
```

2. Spc 函数

格式如下:

Spc(number)

功能: 与 Print 语句一起使用, 对输出进行相对地址定位。

使用说明:

参数 n 是必选的, 指到下一输出项之前需要跳过的空格数。

例如, Tab 函数中的例子也可以如下编程:

```
Print Spc(15);"学号";Spc(8);"姓名";Spc(6);"成绩"
Print Spc(14);String(27,"-")    '输出 27 个减号字符"- "
Print Spc(14);"2007001";Spc(6);"张三";Spc(6);87
Print Spc(14);"2007002";Spc(6);"李四";Spc(6);89
Print Spc(14);"2007003";Spc(6);"王五";Spc(6);96
Print Spc(14);String(27,"-")    '输出 27 个减号字符"- "
```

程序执行后, 得到与 Tab 相同的效果。

3. Format 函数

格式如下:

Format(expression[,format[,firstdayofweek[,firstweekofyear]]])

功能: 根据格式表达式中的指令来格式化表达式。

使用说明:

expression 是必要参数, 可以是任何有效的表达式。

format 是可选参数, 是有效的命名表达式或用户自定义格式表达式。

firstdayofweek 是可选参数。是常数, 表示 1 星期的第 1 天, 见表 3-1。

firstweekofyear 是可选参数。是常数, 表示 1 年的第 1 周, 见表 3-2。

在 VB 中, Format 函数必要时在设置格式之前将字符串转换为数字。如果没有小数部分, Format 显示一个尾随小数点。

VB 在格式设置字符串中支持 4 个部分。这些部分由分号(;)分隔, 分别指定如何设置正、负、零和空值的格式。如果格式字符串的负数部分为空, 则负数会显示一个空字符串。

科学记数法格式设置支持在指数后使用 0 和 # 位占位符。

表 3-1 firstdayofweek 参数

常 数	值	说 明
vbUseSystem	0	使用 NLS API 设置
vbSunday	1	星期日(默认)
vbMonday	2	星期一
vbTuesday	3	星期二
vbWednesday	4	星期三
vbThursday	5	星期四
vbFriday	6	星期五
vbSaturday	0	使用 NLS API 设置

表 3-2 firstweekofyear 参数

常 数	值	说 明
vbUseSystem	0	使用 NLS API 设置
vbFirstJan1	1	从包含 1 月 1 日的那一周开始(默认)
vbFirstFourDays	2	从本年第 1 周开始, 而此周至少有 4 天在本年中
vbFirstFullWeek	3	从本年第 1 周开始, 而此周完全在本年中

例如:

```
Dim MyTime, MyDate, MyStr
```

```
MyTime = #17:04:23#
```

```
MyDate = #January 27, 1993#
```

以系统设置的长时间格式返回当前系统时间语句如下:

```
MyStr = Format( Time, "Long Time")
```

以系统设置的长日期格式返回当前系统日期语句如下:

```
MyStr = Format( Date, "Long Date")
```

```
MyStr = Format( MyTime, "h:m:s") '返回"17:4:23"
```

```
MyStr = Format( MyTime, "hh:mm:ss AMPM") '返回"05:04:23 PM"
```

```
MyStr = Format( MyDate, "dddd, mmm d yyyy") '返回"Wednesday, Jan 27 1993"
```

'如果没有指定格式, 则返回字符串

```
MyStr = Format( 23) '返回"23"
```

'用户自定义的格式

```
MyStr = Format( 5459.4, "##, ##0.00") '返回"5,459.40"
```

```
MyStr = Format( 334.9, "##0.00") '返回"334.90"
```

```
MyStr = Format( 5, "0.00%") '返回"500.00%"
```

```
MyStr = Format( "HELLO", "<") '返回"hello"
```

```
MyStr = Format( "This is it", ">") '返回"THIS IS IT"
```

4. Cls 方法

Cls 可以清除 Form 或 PictureBox 中由 Print 方法和图形方法在运行时所产生的文本或图形, 清除后的区域以背景色填充。设计时使用 Picture 属性设置的背景位图和放置的控件不受 Cls 影响。

格式如下:

```
[<对象名称>].Cls
```

使用说明:

“对象名称”为窗体或图片框, 如果省略“对象名称”, 则清除窗体上由 Print 方法和图形方法在运行时所生成的文本或图形。

例如:

```
Picture.Cls '清除图片框 Picture1 内的图形或文本
```

```
Cls '清除当前窗体显示的内容
```

3.3 窗体

窗体是 VB 系统的一个重要对象。窗体可以显示文本和图形,也可以作为其他控件的“父对象”,所以窗体除了有自已的属性、事件和方法外,还像一个容器一样,可以在上面放按钮、文本框等其他控件。窗体不可见、移动或隐藏时,控件也跟着不可见、移动或隐藏。窗体及上面的控件的一切信息以“.frm"文件存放到磁盘上。

窗体上的属性决定窗体的外观和操作,大多数属性可以用两种方法设置属性:通过属性窗口设置或通过程序代码设置。也有些属性不能用代码设置,只能通过属性窗口设置。

3.3.1 主要属性

- 1) Name 属性:设置窗体名字。
- 2) Caption 属性:设置或返回标题栏内容,常用来标识窗体的功能。
- 3) ForeColor 属:设置窗体上对象的前景色。
- 4) BackColor 属性:设置窗体上对象的背景色。
- 5) Picture 属性:设置窗体所要显示的图片。该属性可以通过 LoadPicture 函数在代码中改变显示的图片。

例如,在属性窗口打开 c:\Documents and Settings\Administrator\My Documents\My Pictures\123.bmp 文件或用下面代码都可以用该“.bmp"文件改变当前窗口的背景:

```
Form1.Picture = LoadPicture ("c:\Documents and Settings\Administrator\My Documents\My Pictures\123.bmp")
```

- 6) Height 和 Width 属性:返回或设置窗体的高度和宽度。为数值型,单位是缇(Twip)。
- 7) Left 和 Top 属性:返回或设置窗体相对于屏幕左上角的位置,为数值型,单位是缇。
- 8) MaxButton 和 MinButton 属性:在属性窗口设置窗体在执行时是否含有最大化和最小化按钮。
- 9) Icon 属性:在属性窗口设置窗体在最小化后呈现的图标。
- 10) Visible 属性:返回或设置窗体在执行时是否可见或是隐藏起来。为 True(默认值)时表示可见;为 False 时表示不可见。
- 11) WindowsState 属性:设置窗体在执行时的状态。
- 12) MDIChild 属性:设置这个窗体是否含有另一个 MDI 子窗体。为 True 时表示有另一个 MDI 子窗体;为 False(默认值)时表示没有另一个 MDI 子窗体。

3.3.2 主要事件

- 1) DblClick 事件:当用户在一个对象上按下并释放鼠标按钮后再次按下并释放鼠标按钮时发生。
- 2) Click 事件:当用户在一个对象上按下并释放鼠标按钮时发生。
- 3) MouseDown 事件:当用户在拥有焦点的对象上按下鼠标按钮时发生。
- 4) Initialize 事件:当应用程序创建一个窗体、MDI 窗体或类的实例时发生。



- 5) Load 事件: 当加载窗体时发生。
- 6) Unload 事件: 当要从屏幕上删除窗体时发生。
- 7) Resize 事件: 当第一次显示一个窗体时或改变一个对象的大小时发生。
- 8) Activate 事件: 当窗体变为活动窗口时发生。
- 9) Deactivate 事件: 当窗体不再是活动窗口时发生。
- 10) QueryUnload 事件: 在关闭窗体或应用程序之前发生。
- 11) Terminate 事件: 当从内存中删除一个窗体、MDI 窗体或类的所有引用时发生。
- 12) ScaleMode 事件: 当使用图形方法或调整控件位置时, 返回或设置一个值, 该值指示对象坐标的度量单位。

3.3.3 主要方法

- 1) Show 方法: 用来显示一个已经装入内存的窗体。其语法如下:

窗体名.Show 模式

其中, “模式”有两个取值: 0(默认值)表示非模式的, 1 表示模式的。

窗体的模式状态是指在焦点可以切换到其他窗体或对话框之前要求用户采取动作。

例如:

Form1.Show 1

窗体的无模式状态是指在焦点可以切换到其他窗体或对话框之前不要求用户采取动作。

例如:

Form1.Show 0

- 2) Hide 方法: 用于隐藏显示在屏幕上的窗体。隐藏窗体时, 将从屏幕上删除窗体, 并将其 Visible 属性设置为 False。用户将无法访问隐藏窗体上的控件, 但是运行中的 VB 应用程序并不卸载它, 仍然可以用代码访问隐藏窗体的控件。该语法如下:

窗体名.Hide

- 3) Cls 方法: 清除窗体上的内容。

- 4) Move 方法: 将窗体移动到相对屏幕左上角的原点(0,0)的位置并且指定窗体宽度和高度。该语法如下:

窗体名.Move left,top,width,height

参数中只有 left 参数是必须的, top、width、height 是可选的。坐标系统或度量单位是在设计时用 ScaleMode 属性设置的。

例如:

Form1.Move 2000,300,600,800

可以将 Form1 窗体移动到距离原点左侧 2000、上面 300, 窗体宽度为 600, 高度为 800。

- 5) Print 方法: 向窗体上打印内容。

例如:

Print "大家好"

执行后, 窗体左上角就出现“大家好”3 个字。

- 6) Refresh 方法: 强制重新绘制窗体及上面的控件。

3.3.4 主要语句

1) Load 语句: 该语句用来将新创建的窗体加载到内存中, 当 VB 加载窗体对象时, 先把窗体属性设置为初始值, 再执行 Load 事件过程。当应用程序开始运行时, VB 自动加载并显示应用程序的启动窗体。其语法如下:

Load 窗体名

例如, Load Form1 功能是把 Form1 窗体加载到内存中。

2) Unload 语句: 用来卸载窗体。在窗体卸载之后, 所有在运行时放到该窗体上的控件都不再是可访问的, 在设计时放到该窗体上的控件将保持不变。对窗体上任何控件的访问都会导致窗体重新加载, 但在重新加载窗体时, 在运行时对这些窗体上的控件及其属性的任何更改将会丢失, 所有对于窗体属性的更改也将会丢失。其语法如下:

Unload 窗体名

3.3.5 窗体的生命周期

由于窗体和控件是可见的对象, 所以它们与其他不可见对象的生命周期不同。例如, 即使释放了对窗体的所有引用, 也不会关闭该窗体。VB 维护整个工程中所有窗体的集合, 只有当窗体卸载时才能从集合中删除该窗体。

通常地, VB 窗体在整个生命周期中要经历 4 个状态: 创建、加载、显示、卸载。

1. 创建状态

创建状态产生 Initialize 事件。

处于这种状态时, 窗体是作为一个对象而存在, 但还没有窗口, 而且它的控件也不存在。虽然该状态可能很短暂, 但任何窗体都要经过该状态。

一旦窗体的 Initialize 事件过程结束, 在不强制加载窗体的情况下, 所能执行的过程只能添加到该窗体代码窗口的 Sub、Function 或 Property 过程中。

2. 加载状态

Load 事件标志加载状态的开始。一旦窗体进入加载状态, 窗体的事件过程中的代码就开始执行。

窗体的 Load 事件过程开始后, 窗体上的所有控件都被创建和加载, 而且该窗体有了一个窗口, 也就是系统为窗体建立了“窗口句柄(hWnd)”和“设备描述体(hDC)”, 尽管该窗口还未显示。

任何窗体只有加载后才能可见。很多窗体自动从创建但不加载状态进入加载但不显示状态, 可以通过调用窗体的 Show 方法进入显示状态。

有时需要窗体保持加载状态, 但不显示, 这是通过窗体的 Load 事件过程中调用窗体的 Hide 方法实现的。任何时候, 只要隐藏了窗体, 它就总是从可见状态回到加载状态。回到加载状态并不重新执行 Load 事件。窗体的 Load 事件过程在窗体的存活期中只运行一次。

3. 显示状态

一旦窗体可见, 用户就能和它交互作用, 这时进入显示状态。任何窗体只有加载后才能显示, 并且窗体在卸载前可以任意隐藏及显示。例如:

Form1.Show

'显示窗体 Form1

Form1.Hide 隐藏窗体 Form1

当一个窗体调用 Hide 方法后, 该窗体变为不可见, 它的属性 Visible 值为 False, 窗体返回到加载状态。用户无法在程序界面上访问隐藏窗体上的控件, 但是对运行的程序来说, 隐藏的控件是可用的, 例如, 当程序运行时, 对加载后隐藏的窗体也可以利用代码对上面的控件修改属性。

窗体在显示状态有两个重要事件: Activate 事件和 Deactivate 事件。当一个窗体变成活动窗体时产生 Activate 事件, 当其他的窗体或应用程序被激活时, 原来活动的窗体产生 Deactivate 事件。

4. 卸载状态

释放内存和资源的唯一办法就是卸载窗体。窗体在卸载时可以是隐藏的, 也可以是可见的。若没有隐藏, 则它保持可见直到卸载完毕。

窗体卸载之前, 先发生 QueryUnload 事件, 后发生的事件为 Unload 事件。如果某些数据希望保存, 可在 QueryUnload 事件过程中提示保存或忽略所做的更改的信息。

卸载窗体时把所有引用设置为 Nothing。这种做法常常会漏掉那些隐含的全局变量引用。如果使用了类名 (正如“属性”窗口中的 Name 属性所示) 来引用窗体, 就等于使用隐含全局变量。为了释放窗体占用的内存, 必须把该变量设置为 Nothing。例如:

```
Set Form1 = Nothing
```

该窗体在撤销前会接收到 Terminate 事件。

总之, 窗体从未加载到加载再到显示, 依次会引发 Initialize 事件、Load 事件和 Activate 事件。窗体卸载时依次引发 Deactivate 事件、QueryUnload 事件、Unload 事件、Terminate 事件。

3.4 常用的内部控件

控件是 VB 系统的重要组成部分。控件不仅提供了自己的事件和方法, 可以编写代码, 完成程序的各种功能, 而且控件属性的设置也会影响到整个程序界面的外观。因此, 应该合理恰当地使用各种不同的控件, 熟练掌握各个控件的属性、事件和方法的使用。

VB 控件在广义上分为 3 类:

- 1) 内部控件, 指 VB 标准工具箱中自带的控件, 例如命令按钮和文本框等控件。
 - 2) ActiveX 控件, 是扩展名为 “.ocx” 的独立文件, 其中包括各种版本 VB 提供的控件 (DataCombo、DataList 控件等) 和仅在专业版和企业版中提供的控件 (例如 ListView、工具栏等), 另外还有许多第三方提供的 ActiveX 控件。使用 ActiveX 控件时, 必须先将控件加载到工具箱中。
 - 3) 可插入的对象, 例如一个包含公司所有员工的列表的 Microsoft Excel 工作表对象, 或者一个包含某工程计划信息的 Microsoft Project 日历对象。使用可插入的对象时, 必须先加载到工具箱中。因为这些对象能添加到工具箱中, 所以可把它们当做控件使用。
- 窗体和窗体上的控件作为对象, 在属性、事件和方法的使用上有很多相似之处。下面从控件的属性、事件和方法 3 方面分别介绍 VB 工具箱中的常用内部控件。

3.4.1 命令按钮

命令按钮的类型名是 `CommandButton`。通常在命令按钮的 `Caption` 属性设置显示按钮功能，并在 `Click` 事件中编写一段程序，当用户用鼠标单击这个按钮或者当前选中该按钮并按下键盘“回车”键时，就会启动这段程序，执行某一特定的功能。大多数 VB 应用程序中都有命令按钮。

1. 常用属性

(1) Caption 属性

设置命令按钮的标题，即命令按钮上显示的文字。界面第一个命令按钮的默认值是 `Command1`。

(2) Cancel 属性

设置命令按钮是否为 `Cancel` 按钮，即当用户按 `Esc` 键时，是否触发它的 `Click` 事件。其值为 `True` 时表示响应 `Cancel` 事件；为 `False` 时表示不响应 `Cancel` 事件，该命令按钮不是 `Cancel` 按钮。

(3) Default 属性

设置命令按钮是否为默认按钮，即当运行程序时，用户按“回车”键时，就激活它。一个窗体上只能有一个命令按钮为默认按钮。当某个命令按钮的 `Default` 设置为 `True` 时，窗体中其他的命令按钮自动设置为 `False`。当命令按钮的 `Default` 设置为 `True` 而且其所在窗体是活动的时，用户可以按“回车”键选择该按钮(激活其单击事件)。

其值为 `True` 时表示该命令按钮为默认按钮；为 `False`(默认值)时表示该命令按钮不是默认按钮。

(4) Enabled 属性

返回或设置命令按钮是否能够对用户产生的事件做出反应。

其值为 `True`(默认值)时表示该命令按钮能被按下以执行特定功能；为 `False` 时表示该命令按钮不能按下来执行特定功能。

(5) Visible 属性

返回或设置一指示对象为可见或隐藏的值。

其值为 `True`(默认值)或非零时表示该命令按钮是可见的；为 `False` 或 0 时表示该命令按钮不可见。

2. 常用事件

常用事件为 `Click` 事件。

满足下面任意一个条件时发生事件：

- 用鼠标的左键单击按钮控件。
- 当命令按钮具有焦点时，按下“空格”键。
- 当窗体带有其 `Default` 属性设置为 `True` 的命令按钮控件时，按下“回车”键。

其基本语法如下：

```
Sub Command_Click([Index As Integer])
```

其中，`Command` 是命令按钮的名称。参数是可选的，若该命令按钮属于一个控件数组，则 `Index` 表示该命令按钮在数组中的下标，否则不需要这一参数。

单击鼠标时命令按钮的 MouseDown 和 MouseUp 事件也被触发, 它们之间区别是事件被触发的时间不一样, 触发的顺序是 MouseDown、Click、MouseUp。

3. 常用方法

常用方法为 Move 方法。其基本语法如下:

命令按钮名.Move left,top,width,height

其中, 参数 left 是必需的, top、width、height 是可选的。功能是相对窗体原点(0,0)改变位置和按钮的宽度、高度。使用方法与窗体的 Move 方法类似。

4. 应用举例

在窗体上放置一个命令按钮, 上面默认的显示文字是 Command1, 单击命令按钮, 将该命令按钮上显示的文字改为“你好”。代码如下:

```
Private Sub Command1_Click()
```

```
    Command1.Caption = "你好"
```

```
End Sub
```

3.4.2 标签

标签控件的类型名是 Label, 该控件一般用于标识窗体上的其他对象或说明。

1. 常用属性

(1) Alignment 属性

设置或返回标签中文本的对齐方式。其中的“值”可以为0、1或2。0(默认值)表示左对齐; 1表示右对齐; 2表示居中。

(2) Caption 属性

设置或返回标签的文本内容。

(3) BackStyle 属性

设置或返回标签的背景样式。

(4) BorderStyle 属性

设置或返回标签的边框样式。该属性的“值”可以为0或1。0(默认值)表示无边框; 1表示单线边框。

(5) AutoSize 属性

设置控件是否能够自动调整大小以显示所有的内容。该属性的“值”为 True 控件自动调整大小, 值为 False 时保持控件大小不变, 超出控件区域的内容被裁剪掉。

2. 常用事件

标签控件的事件有 Click、DblClick、MouseMove、MouseDown、MouseUp 等, 但是一般很少使用。

3. 常用方法

标签的方法中常用的只有 Move, 可以相对窗体移动 Label 位置。与命令按钮该方法相似。

4. 应用举例

现设计一程序, 在窗体上放置两个按钮 Command1、Command2 和一个标签 Label1, 单击按钮可以改变标签控件的边框样式和内容。

窗体上标签程序的控件属性设置见表 3-3。

表 3-3 窗体上标签程序的控件属性设置

控 件	属 性	值
Label1	Caption	现在是无边框 Label 样式
Label1	BorderStyle	0
Command1	Caption	显示无边框
Command2	Caption	显示有边框

代码如下：

```
Private Sub Command1_Click()  
    Label1.BorderStyle = 0  
    Label1.Caption = "现在是无边框 Label 样式"  
End Sub
```

```
Private Sub Command2_Click()  
    Label1.BorderStyle = 1  
    Label1.Caption = "Label 样式是有边框"  
End Sub
```

程序运行后，单击 Command1 显示如图 3-1 所示，单击 Command2 显示如图 3-2 所示。

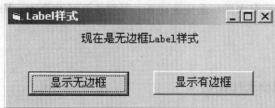


图 3-1 无边框的标签控件

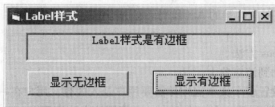


图 3-2 带边框的标签控件

3.4.3 文本框

文本框的类型名是 TextBox，可以显示文本或者接收用户的输入信息。

1. 常用属性

(1) Text 属性



设置或返回控件中显示的文本内容。

(2) Locked 属性

设置或返回控件是否可被编辑，即文本框是否是只读的。当 Locked 属性的属性设置为 True 时，是只读的，不能对文本框内容作变更；否则，不是只读的。

(3) MultiLine 属性和 MaxLength 属性

MultiLine 属性设置文本框是否以多行方式显示文本。设置为 True 时以多行文本方式显示；设置为 False(默认)时以单行方式显示，超出文本框宽度的部分被截除。

MaxLength 属性设置文本框中输入的字符串长度是否有限制。默认值为 0，表示该单行文本框中字符串的长度只受操作系统内存的限制；若设置为大于 0 的数，则表示能够输入的最大字符数目。

(4) PasswordChar 属性

设置或返回是否在控件中显示用户键入的字符。如果该属性设置为某一字符，那么无论 Text 属性值是什么，在文本框中都只显示该字符。另外，要想使该属性有效，MultiLine 属性必须设置为 False。

在对话框中创建一个密码域时可以使用此属性，通常将 PasswordChar 属性值设为星号 (*) (Chr(42))，程序运行后在对话框输入内容后显示出的是星号 “*”。

(5) ScrollBars 属性

设置或返回文本框是否有垂直或水平滚动条。

其中的值可以为 0、1、2、3。0(默认值)表示没有滚动条；1 表示有水平滚动条；2 表示有垂直滚动条；3 表示有水平和垂直滚动条，要想使该属性有效，MultiLine 属性必须设置为 True。

2. 常用事件

(1) Change 事件

当文本框的内容被修改时触发。

(2) KeyPress 事件

当在文本框中按键盘时触发。其基本语法如下：

```
Sub Text_KeyPress(KeyAscii As Integer)
```

其中，KeyAscii 为按键对应的键的 ASCII 码。

3. 常用方法

常用方法为 SetFocus 方法。

功能：将焦点移动到该对象上。

例如，在程序中设置多个文本框控件需要用户录入内容，用户在一个文本框中输入完按“回车”键，焦点自动跳到下一个文本框中，可使用该方法实现。

4. 应用举例

设计一个程序，使一个文本框只能输入数字字符，如果输入非数字字符则发提示音，并将输入文本框的内容放入另一个文本框中。

在窗体上放置左右两个标签，分别为 Label1 和 Label2。在窗体上放置左右两个文本框，分别为 Text1 和 Text2。在属性窗口设置的文本框程序的控件属性见表 3-4。

表 3-4 文本框程序的控件属性

控 件	属 性	值
Label1	Caption	请输入数字
Label2	Caption	您输入的数字是
Text1	Text	""(空值)
Text2	Text	""(空值)
Text2	Locked	True

表 3-4 中, Text2 的 Locked 属性设为 True, 是使它不能在界面被输入内容。

程序代码如下:

```
Private Sub Text1_Change()  
    Text2.Text = Text1.Text  
End Sub  
  
Private Sub Text1_KeyPress(KeyAscii As Integer)  
    If KeyAscii < ("0") Or KeyAscii > Asc("9") Then  
        KeyAscii = 0  
        Beep  
    End If  
End Sub
```

程序运行后如图 3-3 所示。

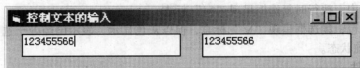


图 3-3 程序运行控制文本的输入后的界面

3.4.4 框架

框架的类型名是 Frame, 它是窗体上的一个矩形区域, 用于放置其他控件以形成独立的一组, 使视觉上不同类型的内容容易区分, 并且可以实现每组的控件同时处于激活或不可用。所以, 框架在实际运用中往往和其他控件一起使用。

将控件分组时, 首先在窗体上放置框架控件, 然后放置框架里面的控件, 这样就可以把框架和里面的控件同时移动。而且, Enable 或 Visible 属性都相同, 使用起来比较方便。

1. 常用属性

(1) Caption 属性

为控件的标题, 起说明作用。

(2) Enable 属性

决定框架和它上面的控件是否可用。如果该值为 True, 框架和它上面的控件全都可用;



反之,该值为 False,框架和它上面的控件都不可用,不必每个控件进行设置了。

(3) Visible 属性

决定框架和它上面的控件是否可见。如果该值为 True,框架和它上面的控件全都可见,反之,该值为 False,框架和它上面的控件同时都不可见。

(4) BorderStyle 属性

设置框架的类型。默认值为 1(有边框),也可设置为 0(无边框)。

2. 常用事件和方法

框架的事件和方法很少使用。

3.4.5 复选框

复选框的类型名是 CheckBox,是用来对一组选项进行选择的控件。一组复选框控件允许用户同时选择多个复选框。

程序运行时,如果是未选中状态,用户用鼠标单击该控件,方框中就会出现一个“√”符号,变为选中状态;如果是选中状态,用户用鼠标单击该控件,变为未选中状态,此时复选框的选项部分是一个空白的小方块。

1. 常用属性

(1) Caption 属性

设置显示标题,用来标识复选框的功能。

(2) Value 属性

设置复选框在执行时的 3 种状态:

0(默认值):表示未复选,处于这种状态的复选框在运行时复选框前没有“√”标志。

1:表示选中,执行时复选框呈现“√”标志。

2:表示灰色,复选框呈现“√”标志,但以灰色显示,表示已经处于选中状态,但禁止用户改变当前状态。

2. 常用事件

Click 事件:当用户在一个复选框上单击鼠标按钮时发生,进行选择或不选该复选框。

3. 应用举例

设计一个程序,可以对窗体上的文本框中的字体进行斜体字和粗体字的设置。在窗体上放置左右两个复选框,分别为 Check1 和 Check2。在窗体上放置一个文本框为 Text1。在属性能窗口设置的复选框程序的控件属性见表 3-5。

表 3-5 复选框程序的控件属性

控 件	属 性	值
Check1	Caption	斜体字
Check2	Caption	粗体字
Text1	Text	大家好

程序代码如下:

```
Private Sub Check1_Click()  
    If Check1.Value Then
```

```

Text1.FontBold = True
Else
Text1.FontBold = False
End If
End Sub

```

```

Private Sub Check2_Click()
If Check2.Value Then
Text1.FontItalic = True
Else
Text1.FontItalic = False
End If
End Sub

```

程序运行后的界面如图 3-4 所示。

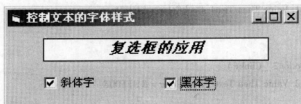


图 3-4 复选框的应用程序运行后的界面

3.4.6 单选按钮

单选按钮的类型名是 `OptionButton`，它也是对一组选项进行选择控件。与复选框允许用户同时做出多个选择不同，一组单选按钮则只能选中其中一个。

程序运行时，单选按钮未选中状态时如果用户用鼠标单击该控件，圆圈中出现一个黑点，变为选中状态，同时，这个组内所有其他单选按钮变为未选中状态。单选按钮的未选中状态，圆圈中是一个空白的。

1. 常用属性

(1) Caption 属性

设置或返回显示标题，用来标识单选按钮的功能。

(2) Value 属性

设置或返回在执行时有两种状态。0(默认值)：表示未选；1：表示选中。

2. 常用事件

Click 事件：当用户在单选按钮上单击鼠标按钮时发生，选择该单选按钮或取消选择。

3. 常用方法

单选按钮的方法很少使用。

注意：单选按钮和复选框的功能类似，都是在多个选项中做出选择。区别是一系列单选按钮中只允许选定其中的一个；而在一系列复选框中却可以选择多个。

4. 应用举例

设计一个程序，可以设置窗体上文本框中的背景颜色。在窗体上放置 3 个单选按钮，分别为 Option1、Option2、Option3。在窗体上放置一个文本框为 Text1。在属性窗口设置的单选按钮程序控件属性见表 3-6。

表 3-6 单选按钮程序的控件属性

控 件	属 性	值
Option1	Caption	红
Option2	Caption	绿
Option3	Caption	蓝
Text1	Text	用单选按钮选择文本框背景颜色

代码如下：

```
Private Sub Option1_Click()
    If Option1.Value Then Text1.BackColor = &HFF&
End Sub

Private Sub Option2_Click()
    If Option2.Value Then Text1.BackColor = &HFF00&
End Sub

Private Sub Option3_Click()
    If Option3.Value Then Text1.BackColor = &HFFF00
End Sub
```

程序运行后的界面如图 3-5 所示。

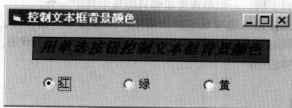


图 3-5 用单选按钮控制文本框背景颜色的程序运行后的界面

3.4.7 列表框

列表框控件的类型名是 ListBox，是向用户提供选择的列表控件，用户可从列表框列出的一组选项中用鼠标选取一个或多个所需的选项。如果有较多的选择项，超出所画的区域而不能一次全部显示时，列表框可以自动加上滚动条。

1. 常用属性

(1) Text 属性

返回当前选中的列表项内容。

(2) ListCount 属性

返回列表框中所有项目的总数。

(3) List 属性

它是一个一维数组，数组中元素的值就是在执行时看到的列表项。设计时可以在属性窗口中输入 List 属性来建立列表项，运行时对 List 数组从 0 到 ListCount - 1 依次取值可以获得列表的所有项目。

(4) Selected 属性

它是一个与 List 数组中的各个元素相对应的一维数组，记录 List 数组中每个项目是否被选取。例如，如果 List(1) 被选取，则 Selected(1) 的值为 True；如果 List(1) 未被选取，则 Selected(1) 的值为 False。

(5) Sorted 属性

设置列表框中的项目是否按字母表顺序排序。Sorted 属性必须在设计时设置，在运行时是只读的。当 Sorted 属性为 True 时，列表框中的项目按字母表顺序排序；当其为 False 时，列表框中的项目不按字母表顺序排序。

2. 常用事件

Click 事件

当用户在列表框的某一个对象上单击鼠标按钮时发生。

3. 常用方法

(1) AddItem 方法

用于将项目添加到列表框中，其语法如下：

列表框名.AddItem, Index

其中，Item 是必需的，是要添加到列表框中的字符表达式；Index 是可选参数，用来指定新项目在列表框中的位置。

如果所给出的 Index 值有效，则 Item 将放置在列表框相应的位置；如果省略 Index，当 Sorted 属性设置为 True 时，Item 将添加到恰当的排序位置，当 Sorted 属性设置为 False 时，Item 将添加到列表的末尾。

(2) RemoveItem 方法

用于从列表框中删除一个项目，其语法如下：

列表框名.RemoveItem Index

其中，Index 用来指定要删除的项目在列表框中的位置。

(3) Clear 方法

删除列表框中的所有项目，其语法如下：

列表框名.Clear

4. 应用举例

设计一个程序，可以对窗体左边的列表框 List1 中的项目进行选择放入右边的列表框 List2 中。在窗体上放置 4 个按钮，功能从上到下分别为一次选一条、全部选择、一次取消一条、全部取消。在窗体上放置左右两个标签 Label1、Label2。在属性窗口设置的列表框程序的控件属性见表 3-7。



表 3-7 列表框程序的控件属性

控 件	属 性	值
Label1	Caption	可选项
Label2	Caption	已选项
Command1	Caption	>
Command2	Caption	>>
Command3	Caption	<
Command4	Caption	<<

代码如下:

```
Private Sub Command1_Click()
```

```
    If List1.ListCount = 0 Then
```

```
        Screen.MousePointer = 0
```

```
        Exit Sub '如果 List1 中没有列表项则退出
```

```
    End If
```

```
    If List1.ListIndex = -1 Then
```

```
        List1.SetFocus
```

```
        List1.Selected(0) = True
```

```
    End If '如果 List1 中没有选中的列表项则选择第一个列表项
```

```
    DoEvents
```

```
    List2.AddItem List1.Text
```

```
    List1.RemoveItem List1.ListIndex
```

```
    '将选择的列表项从 List1 移到 List2
```

```
End Sub
```

```
Private Sub Command2_Click()
```

```
    If List1.ListCount = 0 Then
```

```
        Screen.MousePointer = 0
```

```
        Exit Sub
```

```
    End If '如果 List1 中没有列表项则退出
```

```
    If List1.ListIndex = -1 Then
```

```
        List1.SetFocus
```

```
        List1.Selected(0) = True
```

```
    End If
```

```
    '如果 List1 中没有选中的列表项则选择第一个列表项
```

```
    DoEvents
```

```
    For i = (List1.ListCount - 1) To 0 Step -1
```

```
        List2.AddItem List1.List(i)
```

```
        DoEvents
```

```
    Next i
```

'将 List1 的所有列表项添加到 List2 中

List1. Clear

'删除 List1 中的所有列表项

End Sub

Private Sub Command3_Click()

If List2. ListCount = 0 Then

Screen. MousePointer = 0

Exit Sub

End If

'如果 List2 中没有列表项则退出

If List2. ListIndex = -1 Then

List2. SetFocus

List2. Selected(0) = True

End If

List1. AddItem List2. Text

'如果 List2 中没有选中的列表项则选择第一个列表项

List2. RemoveItem List2. ListIndex

'将选择的列表项从 List2 移到 List1

End Sub

Private Sub Command4_Click()

If List2. ListCount = 0 Then

Screen. MousePointer = 0

Exit Sub

End If

If List2. ListCount = 0 Then Exit Sub

If List2. ListIndex = -1 Then

List2. SetFocus

List2. Selected(0) = True

End If

For i = (List2. ListCount - 1) To 0 Step -1

List1. AddItem List2. List(i)

DoEvents

Next i

List2. Clear

End Sub

Private Sub Form_Load()

List1. AddItem "姓名",0

List1. AddItem "性别",1

List1. AddItem "年龄",2

List1. AddItem "籍贯",3

List1. AddItem "文化程度",4

End Sub

程序运行后的界面如图 3-6 所示。

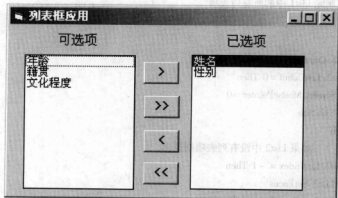


图 3-6 列表框应用程序运行后的界面

3.4.8 组合框

组合框控件的类型名是 ComboBox，是向用户提供选择的列表控件，用户从组合框一次只能选取或输入一个选项的控件，可以从列表框中进行选择，也可以输入信息，所以功能上是文本框和列表框的结合。

1. 常用属性

(1) Style 属性

返回或设置一个用来指示控件的显示类型和行为的值，在运行时是只读的。组合框共有 3 种形式：有下拉式组合框，既可输入又可选择，Style 属性值 = 0；有简单组合框，只能输入不能选择，Style 属性 = 1；有下拉式列表，只能选择不能输入，Style 属性 = 2。

(2) Text 属性

在 Style 属性设置为 0 或 1 时，Text 属性返回或设置编辑框中的文本，Style 属性为 2 时，Text 属性返回列表框中选择的项目。

在设计时，Text 属性的默认值为组合框的名称，可以将 Text 属性设置为空。

Text 属性可在设计时直接在属性窗口中编辑组合框的 List 属性，增加或删除列表项。运行时则要使用 AddItem、RemoveItem 等方法添加、删除列表项，这些方法的使用与列表框控件中相同。

2. 常用事件

(1) Change 事件

当组合框内容发生改变时发生。

(2) Click 事件

当用户在一个组合框上单击鼠标按钮时发生。

3. 常用方法

(1) AddItem 方法

添加一项到组合框控件中。使用方法同列表框的该方法。

(2) Clear 方法

清除组合框的内容。使用方法同列表框的该方法。

(3) RemoveItem 方法

从一个组合框控件中删除一项。使用方法同列表框的该方法。

4. 应用举例

设计一个自动查询程序,功能是对窗体上的 ComboBox 中的项目输入查询单词时,自动从 ComboBox 列表表中找与已输入字母匹配的项目。

代码如下:

```
Private Sub Combo1_Change()  
    Dim sString As String  
    Dim start As Integer  
    start = Combo1.SelStart  
    sString = Left(Combo1.Text, start)  
    For i = 0 To Combo1.ListCount - 1
```

```
        Dim sitem As String  
        sitem = Combo1.List(i)  
        sitem = Left(sitem, start)
```

```
        If sitem = sString Then  
            List1.ListIndex = i  
            List1.Visible = True  
            Exit For  
        End If
```

```
    Next
```

```
End Sub
```

```
Private Sub Combo1_KeyPress(KeyAscii As Integer)
```

```
    If KeyAscii = 13 Then
```

```
        Combo1.ListIndex = List1.ListIndex
```

```
        List1.Visible = False
```

```
    End If
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
    Combo1.AddItem "abc"
```

```
    Combo1.AddItem "acb"
```

```
    Combo1.AddItem "edf"
```

```
    Combo1.AddItem "fff"
```

```
    Combo1.AddItem "xyz"
```

```
    Combo1.AddItem "zqz"
```

```
    '向 Combo1 添加列表项
```

```
    Dim i As Integer
```

```

For i = 0 To Combo1.ListCount - 1 Step 1 '向 List1 添加与 Combo1 相同的列表项
    List1.AddItem Combo1.List(i), i
Next
List1.Visible = False
End Sub

```

自动查询程序运行后的界面如图 3-7 所示。

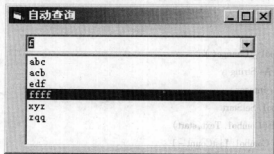


图 3-7 自动查询程序运行后的界面

3.4.9 滚动条

滚动条分为两种类型，即水平滚动条和垂直滚动条，对应的类型名分别是 HScrollBar、VScrollBar。它们的差异是摆放的方向不一样。滚动条与文本框、列表框等一起使用，用来提供简便的定位。

1. 常用属性

(1) Max 属性和 Min 属性

设置滚动条的最大值和最小值，其值介于 -32768 ~ 32767 之间。Max 的默认值为 32767，Min 的默认值为 0。对于水平滚动条来说，最左边为 Min，最右边为 Max；对于垂直滚动条来说，最下面为 Min，最上面为 Max。

(2) Value 属性

表示目前滚动条所在位置对应的值，它是滚动条控件中移动方块位置与最大、最小值换算而得的结果。要取得滚动条的数据，只要取得滚动条的 Value 属性就可以了。

(3) LargeChange 属性

设置用鼠标单击滚动条中间的轴时，每次增减的数值。系统默认数值为 1，用户可以自己修改。

(4) SmallChange 属性

设置用鼠标单击滚动条两边的箭头时，每次增减的数值。系统默认数值为 1，用户可以自己修改。

2. 常用事件

(1) Scroll 事件

只在移动滚动框时被激活，单击滚动箭头或单击滚动条均不能激活该事件。一般可用该事件来监测滚动框的动态变化。

(2) Change 事件

在滚动条的滚动框移动后可以激活，即释放滚动框、单击滚动箭头或单击滚动条时，均会激活该事件。一般可用该事件来获得移动后的滚动框所在的位置值。

3. 常用方法

滚动条的方法不常用。

4. 应用举例

设计一个调色板程序，可以用3个滚动条进行对一个标签背景色调色。

滚动条程序的控件属性设置见表3-8。

表 3-8 滚动条程序的控件属性设置

控 件	属 性	值
Label1	Caption	调色板应用
Label2	Caption	红
Label3	Caption	绿
Label4	Caption	蓝
HScroll1	Min	0
HScroll1	Max	255
HScroll2	Min	0
HScroll2	Max	255
HScroll3	Min	0
HScroll3	Max	255

代码如下：

```
Private Sub HScroll1_Change()  
    Label1.BackColor = RGB(HScroll1.Value, _  
        HScroll2.Value, HScroll3.Value)  
End Sub  
  
Private Sub HScroll2_Change()  
    Label1.BackColor = RGB(HScroll1.Value, _  
        HScroll2.Value, HScroll3.Value)  
End Sub  
  
Private Sub HScroll3_Change()  
    Label1.BackColor = RGB(HScroll1.Value, _  
        HScroll2.Value, HScroll3.Value)  
End Sub
```

滚动条程序运行后的界面如图3-8所示。

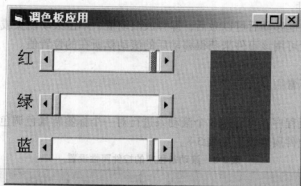


图 3-8 滚动条程序运行后的界面

3.4.10 计时器

计时器的类型名是 Timer，用于定时。该控件程序运行时不可见。

1. 常用属性

(1) Enabled 属性

决定计时器控件是否开始计时。若设置为 True (默认值)，表示启动计时器开始计时；否则，表示暂停计时器的使用，在需要启动计时器时再将 Enabled 属性设置为 True 即可。

(2) Interval 属性

设置两个计时器事件之间的时间间隔。设置时以毫秒 (ms) 为单位，设置的范围是 0 ~ 65535ms。若将 Interval 属性值设置为 1000ms，即将时间间隔设置为 1s，则每隔 1s 就会执行一个计时器时间。系统默认值为 0，不计时。

2. 常用事件

计时器的主要事件就是 Timer 事件。当 Enabled 属性值为 True 时，在每隔 Interval 指定的时间间隔就执行一次该事件过程。

3. 常用方法

该控件无方法。

4. 应用举例

设计一个程序，可以显示当前时间，并且每秒用扬声器发提示音一声。鼠标单击窗体时关闭程序。

代码如下：

```
Private Sub Form_Click()  
    End  
    Unload Me  
End Sub
```

```
Private Sub Form_Load()  
    Form1.Caption = "显示时间"  
    Timer1.Interval = 1000
```

End Sub

```
Private Sub Timer1_Timer()
```

```
    Label1.Caption = Time
```

```
    Beep
```

```
End Sub
```

定时器程序运行后界面如图 3-9 所示。

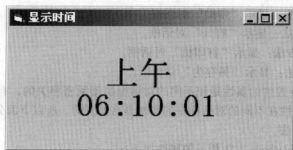


图 3-9 定时器程序运行后的界面

3.5 ActiveX 控件

ActiveX 控件是一个提供特定功能的二进制对象，在对象内部封装了数据和对数据的处理方法。ActiveX 控件采用一种开放式的 Internet 对象引用技术，它不是某种特定编程语言的产物，因此，使用不同编程语言编写的 ActiveX 控件都可以得到广泛的应用。

VB 中的 ActiveX 控件以“.ocx”为扩展名保存在文件中，需要时应该将它调入工具箱中才能使用。

将 ActiveX 控件调入 VB 标准工具箱的方法是：单击“工程”菜单的“部件...”命令，从“部件”对话框里打开“控件”选项卡，显示 ActiveX 控件的列表。在列表中找到需要的控件，勾选该控件前面的复选框。如果在控件列表中没有，单击“浏览”按钮，打开“添加 ActiveX 控件”对话框，找到需要的“.ocx”文件后，单击“打开”按钮，选择的 ActiveX 控件加入到 ActiveX 控件的列表中并且勾选对应的复选框即可。

进入工具箱后，使用 ActiveX 控件的方法和使用 VB 内部控件方法相同。

将 ActiveX 控件从 VB 标准工具箱删除的方法是：单击“工程”菜单的“部件...”命令，打开显示 ActiveX 控件的列表，在列表中找到该控件，去掉控件前面的复选框的勾。

Windows 公共控件是 VB 系统给程序员提供用来建立具有 Windows 外观风格界面的 ActiveX 控件，其中包括工具栏、状态栏、进度条等多个控件，在使用之前，必须先把这些控件加到工具箱中。方法是：执行菜单“工程”中的“部件...”命令，在“部件”选项卡中选择“Microsoft Windows Common Controls 6.0”，就把 Windows 公共控件包括的控件都放在工具箱中了。

3.5.1 通用对话框

通用对话框的类型名是 `CommonDialog`，该控件在程序运行后为不可见。

通用对话框包括“颜色”、“字体”、“帮助”、“打开”、“打印机”、“另存为”对话框，每一种对话框对应一种打开方法：

- 1) `ShowColor` 方法：显示“颜色”对话框。
- 2) `ShowFont` 方法：显示“字体”对话框。
- 3) `ShowHelp` 方法：显示“帮助”对话框。
- 4) `ShowOpen` 方法：显示“打开”对话框。
- 5) `ShowPrinter` 方法：显示“打印机”对话框。
- 6) `ShowSave` 方法：显示“另存为”对话框。

而 `CommonDialog` 控件的属性是和不同的对话框类型紧密相关的，有些属性只适用于某一类对话框，有些属性在不同的对话框中的属性是有差别的，所以下面分类列出了和不同对话框相关联属性的用法。

与 `ShowOpen`、`ShowSave` 方法相关的属性如下：

1) `FileName` 属性：返回或设置所选文件的路径和文件名，如果在使用 `Show` 方法以前使用 `FileName` 属性，则设定了对话框的默认文件名；如果是在以后使用则返回选择的文件名。

使用语法如下：

```
CommonDialog.FileName[ = pathname]
```

2) `Filter` 属性：返回或设置在对话框的类型列表框中所显示的过滤器（也就是限定打开或保存为的文件类型），它的使用语法如下：

```
object.Filter[ = 描述文字 1 | 过滤标示 1 | 描述文字 2 | 过滤标示 2]
```

其中，描述文字为任意文字，而过滤标示则采用“*. 文件后缀”（例如“*.bmp”）的格式，描述文字和过滤标示之间用“|”隔开。

3) `DefaultExt` 属性：为该对话框返回或设置默认的文件扩展名，也就是当没有指定打开或保存的文件类型时，按 `DefaultExt` 属性所设置的扩展名为默认值。

4) `FileTitle` 属性：返回要打开或保存文件的名称（没有路径）。

5) `InitDir` 属性：返回或设置初始文件目录。

6) `FilterIndex` 属性：返回或设置“打开”或“另存为”对话框中一个默认的过滤器。指出文本框中现在显示的项目。

与 `ShowFont` 方法相关的属性如下：

1) `Flags` 属性：用于设置初始字体。

2) `Color` 属性：选定的颜色。为使用此属性，必须先将 `Flags` 属性设置为 `cdlCFEffects`。

3) `FontBold` 属性：是否选定“粗体”。

4) `FontItalic` 属性：是否选定“斜体”。

5) `FontStrikethru` 属性：是否选定删除线。

6) `FontUnderline` 属性：是否选定下画线。

7) `FontName` 属性：选定的字体名称。

8) FontSize 属性: 选定的字体大小。

这些属性的用法是直接引用, 比如要根据“字体对话框”返回的值设置文本框的字体, 则直接采用如下语句:

```
Text.Font = CommonDialog.FontName
```

与 ShowColor 方法相关的属性如下:

1) Flags 属性: 用于设置颜色对话框中初始颜色。

2) Color 属性: 选定的颜色。为使用此属性, 必须先将 Flags 属性设置为 cdlCFEffects。

与 ShowHelp 方法相关的属性如下:

1) HelpCommand 属性: 返回或设置需要的联机帮助的类型。

2) HelpFile 属性: 确定帮助文件的路径和文件名。语法如下:

```
object.HelpFile[ = filename]
```

3) CancelError 属性: 它设置当选取“取消”按钮时是否认为出错, 使用的语法如下:

```
CommonDialog.CancelError[ = boolean] (boolean 指布尔型变量)
```

如果把它设为 True, 则当使用者选取了“取消”按钮时程序会返回一个 cdlCancel 错误, 通过捕捉这个错误并加以处理, 就能避免程序的出错。具体的使用可在例 3-2 中看到实例。

【例 3-1】 设计一个程序, 分别打开通用对话框中的 6 种对话框。

新建一个工程, 在窗体上放置 6 个文本框、6 个按钮, 1 个通用对话框控件。如图 3-10 所示。

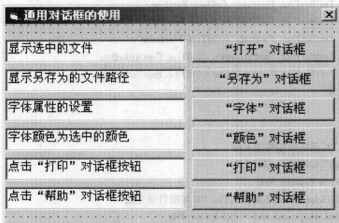


图 3-10 通用对话框程序

代码如下:

```
Private Sub CmdColor_Click()  
    On Error Resume Next  
    CdlTest.CancelError = True  
    CdlTest.Flags = cdlCCRGBInit  
    CdlTest.ShowColor  
    If Err = cdlCancel Then Exit Sub
```

```
        TextBoxColor.ForeColor = CdlTest.Color
    End Sub

'当“字体”对话框按钮被按下时
Private Sub CmdFont_Click()
    On Error Resume Next
    '当用户按下“取消”按钮,返回一个错误信息,这样可以对其进行控制
    CdlTest.CancelError = True
    '此句必须要
    CdlTest.Flags = cdlCFBoth + cdlCFEffects
    '显示“字体”对话框
    CdlTest.ShowFont
    '出现“取消”错误时,跳出
    If Err = cdlCancel Then
        Exit Sub
    Else
        '将 TextBox 的字体属性根据“字体”对话框的变化作相应设置
        '如果用户选择了字体才将字体改变,避免字体为空的错误
        If CdlTest.FontName < > "" Then
            TextBoxFont.FontName = CdlTest.FontName
        End If
        TextBoxFont.FontSize = CdlTest.FontSize
        TextBoxFont.FontBold = CdlTest.FontBold
        TextBoxFont.FontItalic = CdlTest.FontItalic
        TextBoxFont.FontStrikethru = CdlTest.FontStrikethru
        TextBoxFont.FontUnderline = CdlTest.FontUnderline
    End If
End Sub

'当“帮助”对话框按钮被按下时
Private Sub CmdHelp_Click()
    On Error Resume Next
    '设置 HelpCommand 属性,显示 VB 帮助目录主题
    CdlTest.HelpCommand = cdlHelpForceFile
    '指定帮助文件
    Dim fullpath As String
    If Right(App.Path, 1) = "\" Then '若 App.Path 为根目录
        fullpath = App.Path + "test.hlp"
    Else
        fullpath = App.Path + "\" + "test.hlp"
    End If
    '上面是得到应用程序所在路径的小技巧
```

```

CdITest.HelpFile = fullpath
'显示“帮助”对话框
CdITest.ShowHelp
End Sub

```

'当“打开”对话框按钮被按下时

```

Private Sub CmdOpen_Click()
'出现错误时跳到下一语句
On Error Resume Next
CdITest.CancelError = True
'属性 DialogTitle 是要弹出的对话框的标题
CdITest.DialogTitle = "打开文件"
'默认的文件名为空
CdITest.FileName = ""
'属性 Filter 是文件过滤器,返回或设置在对话框的类型列表框中所显示的过滤器
'语法 object.Filter [ = 文件类型描述 1 |filter1 | 文件类型描述 2 |filter2,... ]
CdITest.Filter = "文本文件(.txt)|*.txt"
'Flags 属性的用法依据不同的对话框而变,详细使用需要查找联机帮助手册
CdITest.Flags = cdIOFNCreatePrompt + cdIOFNHideReadOnly
CdITest.ShowOpen
If Err = cdICancel Then Exit Sub
TextBoxOpen.Text = CdITest.FileName
End Sub

```

'当“打印”对话框按钮被按下时

```

Private Sub CmdPrint_Click()
On Error Resume Next
CdITest.CancelError = True
'显示“打印”对话框
CdITest.ShowPrinter
If Err = cdICancel Then Exit Sub
End Sub

```

'当“保存”对话框按钮被按下时

```

Private Sub CmdSave_Click()
On Error Resume Next
CdITest.CancelError = True
CdITest.DialogTitle = "保存文件"
CdITest.FileName = ""
'解释见上面
CdITest.Filter = "文本文件(*.txt)|*.txt"
CdITest.Flags = cdIOFNCreatePrompt + cdIOFNHideReadOnly

```

```

Cd1Test.ShowSave
If Err = cdlCancel Then Exit Sub
TextBoxSave.Text = Cd1Test.FileName
End Sub

```

3.5.2 进度条控件

进度条控件的类型名是 `ProgressBar`，可用于图形显示事件的进程。该控件的边框在事件进行过程中逐渐被充满。

1. 常用属性

(1) Value 属性

返回或设置对象的值。

(2) Max 属性

返回或设置当滚动框处于底部或最右位置时，一个滚动条位置的 `Value` 属性最大设置值。对于进度条控件，它返回或设置其最大值。

(3) Min 属性

返回或设置当滚动框处于顶部或最左位置时，一个滚动条位置的 `Value` 属性最小设置值。对于进度条控件，它返回或设置其最小值。

(4) BorderStyle 属性

返回或设置对象的边框样式。

2. 举例

设计一个程序，使用进度条控件和定时器，每 100s 进度条填满一次，发出提示音后，再重新开始。

在窗体上放置进度条控件和定时器控件，其中进度条控件 `Min = 0`，`Max = 100` (控件默认值)。

代码如下：

```

Private Sub Form_Load()
    Timer1.Interval = 1000
    ProgressBar1.Height = 300
End Sub

Private Sub Timer1_Timer()
    Static Counter As Integer
    Counter = Counter + 1
    ProgressBar1.Value = Counter
    If Counter >= 100 Then
        Counter = 0
        Beep
    End If
End Sub

```

进度条程序运行后的界面如图 3-11 所示，并且每秒计算机响一次，100s 进度条满重新开始。

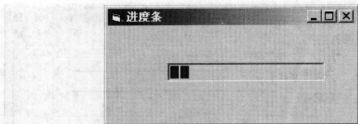


图 3-11 进度条程序运行后的界面

3.5.3 图像列表控件

图像列表控件的类型名是 `ListImage`，是一个向其他控件提供图像的资料中心，它包含了一组 `ListImage` 对象即一组图像的集合，该集合中的每个对象都可以通过其索引或关键字被其他控件所引用，但控件本身并不能单独使用。程序运行时该控件不可见。

1. 常用属性

(1) `ImageHeight` 属性和 `ImageWidth` 属性

返回或设置图像列表控件中 `ListImage` 对象的高度和宽度。可以在添加第一个 `ListImage` 对象之前预先设置 `ImageHeight` 和 `ImageWidth` 属性。如果没有预先设置，则第一个添加到集合中的 `ListImage` 对象保持原图大小，其后添加的所有 `ListImage` 对象都被强制与第一个 `ListImage` 对象大小相同。

(2) `MaskColor` 属性

返回或设置在图像列表控件的图形操作中透明的颜色。也就是设置一种颜色，使图像列表控件的图形中的此种颜色变为透明。

(3) `UseMaskColor` 属性。

决定是否能在图像列表控件中使用 `MaskColor` 属性。

2. 常用事件

(1) `Add` 方法

在运行时，可以用 `Add` 方法，给图像列表控件添加图像。

使用语法如下：

```
Add(index, key, picture)
```

说明：

`index` 是可选的参数。它是一个整数，指定了要插入的 `ListImage` 对象的位置。如果没有指定 `index`，`ListImage` 对象将被添加到 `ListImage` 集合的末尾。

`key` 也是可选的参数。它是用来标识 `ListImage` 对象的唯一字符串。图像列表控件用该值来检索某个特定的 `ListImage` 对象。

`picture` 是必需的参数。它指定了欲添加到集合中的图片。

在设计时要给图像列表控件添加图像，可以用图像列表控件属性页对话框，如图 3-12 所示，具体方法见选项卡控件和工具栏控件中的方法。

(2) `Overlay` 方法

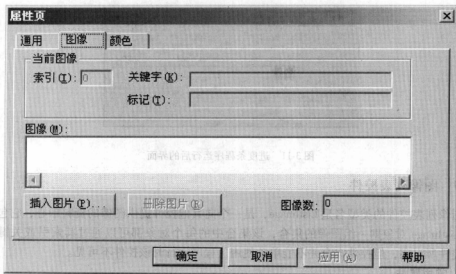


图 3-12 图像列表控件属性页对话框

Overlay 方法可以从两个不同的图像出发创建一个复合图像。该方法从一个 ListImage 集合绘制一幅图像，叠加在另一幅上面，并返回结果。

它的语法如下：

Overlay(index1, index2)

index1 参数是必需的。它是一个整数(Index 属性)或唯一的字符串(Key 属性)，指定了将被叠加的图像。

index2 参数也是必需的。它指定了将被绘制在由 index1 指定的对象上的图像。该图像中与 MaskColor 属性相匹配的颜色被设置成透明的。如果没有任何颜色与之匹配，该图像将不透明地绘制在其他图像上。

3.5.4 选项卡控件

选项卡控件的类型名是 TabStrip，功能与笔记本的分页签差不多。使用后就能够将应用程序中的窗口或对话框的同一区域定义为多页。

1. 用属性页窗口设置常用属性

将选项卡控件放在窗体上后，选中该控件，单击鼠标右键选择“属性”命令，进入选项卡控件属性页窗口，如图 3-13 所示。

(1) 设置样式

在属性页窗口里选择“通用”选项卡，在“样式”列表框里选择 Tabs 或 Buttons 样式。

(2) 设置标题和提示文本

在属性页窗口里选择“选项卡”选项卡，在“标题”框里输入标题。在属性页窗口里选择“选项卡”选项卡，在“工具提示文本”框里输入提示文本。

(3) 增减选项卡总数

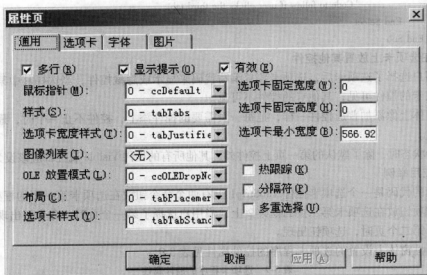


图 3-13 选项卡控件属性页

选择“选项卡”选项卡，单击“插入选项卡”按钮增加选项卡，单击“删除选项卡”按钮删除选项卡。

(4) 在选项卡上显示图形

首先在窗体里拖入一个图像列表控件，选中后单击鼠标右键，单击“属性”命令，打开属性页窗口。在“图像”选项卡中，单击“插入图片”按钮，在“选定图片”对话框里选择想使用的位图或图标，然后单击“打开”按钮，即添加一个图片。重复此步，直到为每个需要的选项卡添加图形。单击“确定”按钮。现在“图像列表”控件里存储了需要的图形。

然后选择该选项卡，进入属性窗口并选择“通用”选项卡，在图像列表框里选择ImageList1。

选择“选项卡”选项卡，单击紧挨“索引”框的向左或向右箭头以选择一个索引号。在“图像”框里输入一个数，输入为1，则显示刚才图像列表控件里的第一个图形，输入为2，则显示刚才图像列表控件里的第二个图形。

最后单击“确定”按钮，结束属性设置。

2. 常用事件

选项卡控件的事件 Click 可用类似下面的代码来识别用户在哪个项目上单击：

```
Private Sub TabStrip1_Click()  
    Select Case TabStrip1.SelectedItem.Index  
        Case 1  
            ' Code to follow if user clicks the first tab  
        Case 2  
            ' Code to follow if user clicks the second tab  
        Case 3
```



```
' Code to follow if user clicks the third tab
```

```
End Select
```

```
End Sub
```

3. 在选项卡上放置其他控件

在用户选择了选项卡后，选项卡要求编写代码来显示或隐藏控件，当选择其中某一选项卡时，该卡的控件可见，其他卡的控件不可见。

在窗体上像添加普通控件一样，把每一页需要的控件添上，控件不止一个时，最好用框架来分组。

初始状态时，除了默认的第一页上控件外，其他所有的控件 Visible 属性都应该设为 False。

4. 应用举例

下面的代码是一个选项卡控件 TabStrip1 上有两个命令钮放在选项卡第一页的框架 1 上，有两个选项钮放在选项卡第二页的框架 2 上，当用户选择了第一个页时，命令钮出现，当用户选择了第二个页时，选项钮出现。

在属性窗体上设置的选项卡程序的控件属性见表 3-9。

表 3-9 选项卡程序的控件属性

控 件	属 性	值
Form1	Caption	TabStrip
Frame1	Visible	True
Frame2	Visible	False

选中 TabStrip1 控件，单击鼠标右键，单击“属性”命令，在“选项卡”页，索引 1 的标题填入“第一页”；单击插入选项卡按钮，进入索引 2 的标题填入“第二页”。

代码如下：

```
Private Sub Form_Load()
```

```
Dim i
```

```
Frame2. Visible = False
```

```
Frame1. Visible = True
```

```
End Sub
```

```
Private Sub TabStrip1_Click()
```

```
Select Case TabStrip1.SelectedItem.Index
```

```
Case 1
```

```
Frame2. Visible = False
```

```
Frame1. Visible = True
```

```
Case 2
```

```
Frame2. Visible = True
```

```
Frame1. Visible = False
```

```
End Select
```

```
End Sub
```

程序运行后，单击选项卡的第一页和第二页的界面如图 3-14 和图 3-15 所示。



图 3-14 单击选项卡第一页的界面

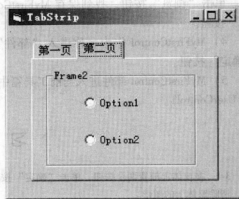


图 3-15 单击选项卡第二页的界面

3.6 第三方控件的使用

第三方控件就是由别的公司或个人开发的而不是 VB 系统提供的控件，这些控件提供或补充了 VB 系统中控件所没有的功能和特色。使用第三方控件的优点是可以加快开发速度，提高开发效率；缺点是有的第三方控件有兼容性问题。

VB 系统给程序员提供了很多基本的控件，这些控件包含在 VB 中自带的 ActiveX 控件库中，大大丰富了 VB 的功能。然而，只有这些基本的控件在很多时候都不能满足实际需要，因此，还需要更多的能够实现其他特有功能的控件。

第三方控件以 ActiveX 控件形式存在。在 VB 中不但可以使用现有的 ActiveX 控件，并且可以利用 VB 创建自己的 ActiveX 控件。

创建一个新 ActiveX 控件，一般应遵循的步骤如下：

- 1) 确定控件将要提供的功能。因为 ActiveX 控件类似于一个独立的对象，所以需要明确：这个对象的目的；希望它在屏幕上有什么样的外观；使用此控件时，需要什么属性、方法和事件用于应用程序中。
- 2) 设计控件的外观。
- 3) 设计控件的接口，即属性、事件和方法。
- 4) 创建由控件工程和测试工程组成的工程组。
- 5) 通过把控件和或代码添加到 UserControl 对象中来实现控件的外观。
- 6) 实现控件的接口和功能。
- 7) 编译控件部件（“.ocx”文件）。

例如，要在 VB 程序中使用第三方控件 MyFirstControl.ocx 的 ActiveX 控件，方法如下：

- 1) 将 MyFirstControl.ocx 复制到 Windows 文件夹中的 system 文件夹中，或通过命令行命令 Regsvr32 进行注册。
- 2) 打开或新建需要使用该控件的 VB 的工程文件。
- 3) 单击“工程”菜单中“部件...”命令，打开“部件”对话框，选择“控件”选项



卡。单击“浏览”按钮，找到保存 MyFirstControl.ocx 文件的位置，选中该文件，单击“打开”按钮。

4) MyFirstControl 作为名称进入“部件”列表中，在该部件前面的复选框选中，单击“确定”按钮。

5) MyFirstControl 部件进入当前工具箱中，名称是这个控件保存的控件文件(.ctl)的文件名 UserControl1。

习 题

3-1 在窗体上放置两个按钮，单击“确定”按钮，在窗体上输出图 3-16 所示图形，单击“清除”按钮，清除窗体上的图形。

3-2 编写一个程序进入登录窗口，单击“确定”按钮，如果用户名为张三，密码为 123 则输出成功登录提示，否则，输出登录失败提示，如图 3-17 所示。



图 3-16 习题 3-1 图

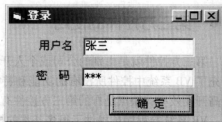


图 3-17 习题 3-2 图

3-3 编写一个程序，在窗体中放置一个内容为“大家好”的标签控件，标签背景颜色是绿色，标签字体的颜色为红色。

3-4 编写一个“选择旅游地区”程序，用户可以在窗体上选择自己想去的旅游地区。假定被选地区有加拿大、新加坡、泰国，用户可以进行单项或多项的选择，然后在一个文本框中显示所选择的旅游地区。

3-5 编写一个程序，在窗体中放置一个文本框控件，3 个单选按钮分别对应宋体、黑体、楷体，选择单选按钮后文本框内显示当前选择的是哪种字体，同时字体变为该字体。

3-6 电子倒计时器。先由用户给定倒计时时的初始秒数，然后开始倒计时，当计到 0 分 0 秒时，通过消息对话框显示“倒计时结束”。

3-7 编写一个程序，在窗体中放置一个水平滚动条和一个文本框控件，文本框中显示水平滚动条的当前值，设置水平滚动条的 Min 属性为 0，Max 属性为 100，SmallChange 属性为 2，LargeChange 属性为 20，默认的 Value 属性为 90。

3-8 编写一个程序，在窗体中放置的“打开文件”按钮，单击该按钮进入 c:\Windows 文件夹，打开文件类型为“.doc”。

3-9 编写一个程序，在窗体中放置一个文本框和工具栏的 3 个按钮，分别是加粗、倾斜和下划线，程序运行后，按工具栏上的按钮，改变文本框的字体变为加粗、倾斜、或下划线。

3-10 掌握第三方控件的开发和使用方法。利用该方法设计一个 ActiveX 控件，功能是显示当前日期、时间和星期。

第4章 程序控制结构

计算机程序的执行和控制流程有顺序结构、分支结构和循环结构3种基本的结构,这3种基本结构可以组成任何功能复杂的程序。

在传统的面向程序的程序设计中,程序的结构无论从宏观到微观都由这3种结构组成。

可视化程序设计方法中增加了事件驱动机制,使程序按照事件发生的次序随机执行,而不是按照编程时就定义好的顺序执行,当某个事件发生时,程序将找到相应的事件处理程序来处理事件。这种机制虽然将每个事件的处理过程划分为比较小的子过程,但是对于具体每个事件处理过程内部而言,也是由这3个基本结构组成。因此,本章介绍计算机程序控制的3种基本结构。

4.1 顺序结构程序

所谓顺序结构,就是按照语句出现的顺序一条一条地执行。这种语句的特点是根据语句的先后次序,依次向下执行。

在第3章介绍的赋值语句和输入、输出语句都是顺序结构中的常用语句,下面介绍另外几种基本顺序结构的语句和方法。

4.1.1 End 语句

End 语句功能是用来结束程序。

例如,在窗体上放置一个“退出”按钮,在 Click 事件中使用 End 语句。

```
Private Sub Command1_Click()
```

```
End
```

```
End Sub
```

程序运行后,单击“退出”按钮,结束程序。

4.1.2 Rem 语句

为了提高程序的可读性,通常在程序的适当位置上加一些注释。Rem 语句可以用来在程序中包含注释。

VB 中注释语句的前缀是一个单引号(')或者是保留字 Rem。

使用保留字 Rem 方法时,如果注释语句是单独一行,以 Rem 开头,后面写入内容即可,如果注释在语句之后,要用冒号(:)隔开。例如:

```
Rem 这是我的注释例子程序
```

```
InputBox("请输入数据"): Rem 这是输入语句
```

使用单引号(')时,只要直接在单引号(')的后面写入内容即可,例如:

```
InputBox("请输入数据")'这是输入语句
```



说明:

1) 注释语句是非执行语句, 仅对程序起注释说明作用。它不被解释和编译, 因此, 如果用 GoTo 转向使用行号或者行标签指明的一个 Rem 语句行, 程序会从该 Rem 语句下面的第一条可执行语句继续执行。

2) 注释语句通常可以作为一段程序的说明放在程序的开头, 或者作为某语句的说明放在该语句的后面。

3) 如果使用多行注释, 每一行都要使用 Rem 或(')。例如:

下面的程序完成如下功能:

'1、打开数据库

'2、对内容进行排序

'3、关闭数据库

4) 注释用来描述程序中的关键部分和整个程序的功能。一般来讲, 程序的总体功能注释放在程序的开头部分。

5) 注释语句还可以用于调试程序。程序员在编写代码时暂时不能确定是否需要的语句, 可加上 Rem 或('), 等确定是否需要该语句时, 再去掉注释符号或直接删除该语句。

4.2 分支结构程序

当程序执行时遇到需要根据不同的情况做出不同的选择, 从而执行不同的操作时, 可以使用分支结构。VB 系统提供了多个语句、函数和控件来实现分支结构。

4.2.1 If 语句

分支结构中具有两个分支选项的情况通常使用 If 语句。If 语句有以下几种格式:

1. If...Then 单行结构

语句格式如下:

If 条件 Then 语句 1 [Else 语句 2]

语句 1 在块形式中是可选参数, 但是在单行形式中, 且没有 Else 子句时, 则为必选参数。一条或多条以冒号分开的语句, 它们在条件为 True 时执行。

语句 2 在条件为 False 时执行。

其中, 条件为必要参数, 可以是一个或多个数值表达式或字符串表达式, 一般为条件表达式。条件表达式的值只有两种情况: 真或假 (对应数值表达式取值为非零或零)。如果条件为 Null, 则条件会视为 False。

满足条件有两种可能: 条件表达式计算后的结果为非零值, 或者逻辑表达式结构为真。格式中的 Else 部分是可以省略的。

功能: 对条件进行判断后, 根据所得的不同结果进行不同的操作。当条件成立时, 执行 Then 后面的语句 1, 执行完后再执行整个 If 语句后的语句; 当条件不成立时, 若存在 Else 部分, 则执行 Else 后的语句 2, 再执行整个 If 语句后的语句, 否则就直接执行整个 If 语句后的语句。

例如, 如果已知两个变量 a 和 b, 将 a 和 b 中的比较大的数赋值给变量 a:

If a < b Then a = b

又如, 求 a 和 b 两个变量中的比较大的数赋值给 Max 变量:

If a > b Then Max = a Else Max = b

注意: If 条件 Then 语句 1 [Else 语句 2] 单行格式不用 Endif 结束。

【例 4-1】 设计一个程序, 功能是输入 3 个数, 求出最大数。

新建一个工程, 在窗体上面放置 4 个标签控件、4 个文本框控件、1 个命令按钮。求最大数程序控件属性设置见表 4-1。

表 4-1 求最大数程序控件属性设置

控 件	属 性	值	控 件	属 性	值
Label 1	Caption	a =	Label 4	Caption	最大数 =
Label 2	Caption	b =	Command 1	Caption	求最大数
Label 3	Caption	c =			

程序代码如下:

```
Private Sub Command1_Click()
    Dim a As Integer, b As Integer
    Dim c As Integer, m As Integer
    a = Val(Text1.Text)
    b = Val(Text2.Text)
    c = Val(Text3.Text)
    If a > b Then
        m = a          'm 用来存放较大值
    Else
        m = b
    End If
    If c > m Then m = c
    Text4.Text = m
End Sub
```

求最大数程序运行后的工作界面如图 4-1 所示。

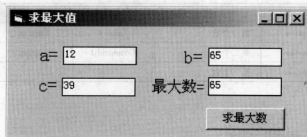


图 4-1 求最大数程序运行后的工作界面

在 If...Then 单行结构中, 不管是哪种结果, 操作部分都必须是单个语句。如果执行多



条语句，语句必须在同一行上并且以冒号分开。

例如：

```
If A > 10 Then A = A + 1; B = B + A; C = C + B
```

因为这种方式要求所有语句必须在同一行，限制比较多，所以如果分支操作执行的语句比较多，通常采用下面所述多行格式 If 语句结构。

2. 多行格式 If 语句

多行格式又分为 3 种格式。

(1) 格式 1

语句格式如下：

```
If 条件 Then
```

```
    语句体
```

```
End If
```

功能：当条件成立时，执行 Then 后面的语句体中的全部语句，执行完后跳出整个 If 语句体，执行 If 语句体后的语句；当条件不成立时，直接执行整个 If 语句体后的语句。

【例 4-2】设计一个求两个数中最小数的函数。

代码如下：

```
Private Function MinValue(x As Integer,y As Integer) As Integer
```

```
    If x < y Then
```

```
        MinValue = x
```

```
    End If
```

```
    If x > = y Then
```

```
        MinValue = y
```

```
    End If
```

```
End Function
```

【例 4-3】设计一个程序，功能是在窗体的文本框中输入 3 个数，单击“排序”按钮可以对 3 个数进行排序，结果显示在另外一个文本框中。

在程序的窗体上放置文本框 Text1、Text2、Text3、Text4，标签 Label1、Label2、Label3、Label4 和命令控件 Command1，排序程序控件属性设置见表 4-2。

表 4-2 排序程序控件属性设置

控 件	属 性	值	控 件	属 性	值
Label 1	Caption	第一个数	Label 4	Caption	排序结果
Label 2	Caption	第二个数	Command 1	Caption	排序
Label 3	Caption	第三个数			

代码如下：

```
Private Sub Command1_Click()
```

```
    a = Val(Text1.Text)
```

```
    b = Val(Text2.Text)
```

```
    c = Val(Text3.Text)
```

```

If a < b Then          '本条件语句实现 a > = b
    t = a
    a = b
    b = t
End If

If a < c Then          '本条件语句实现 a > = c
    t = a
    a = c
    c = t
End If

If b < c Then          '本条件语句实现 b > = c
    t = b
    b = c
    c = t
End If

Text4.Text = a & ", " & b & ", " & c

End Sub

```

排序程序运行后的工作界面如图 4-2 所示。

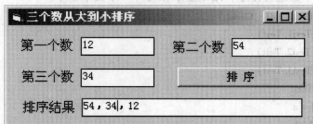


图 4-2 排序程序运行后的工作界面

(2) 格式 2

语句格式如下:

If 条件 Then

语句体 1

Else

语句体 2

End If

功能: 当条件成立时, 执行 Then 后面的语句体 1 中的全部语句, 执行完后跳出整个 If 语句体, 执行 If 语句体后的语句; 当条件不成立时, 则执行 Else 后的语句体 2 中的全部语句, 再执行整个 If 语句体后的语句。语句体 1 和语句体 2 可以使单个语句, 也可以是多个语句。

(3) 格式 3

语句格式如下:

If 条件 1 Then



```
    语句体 1
ElseIf 条件 2 Then
    语句体 2
ElseIf 条件 3 Then
    语句体 3
...
Else
    语句体 n
End If
```

功能：当条件 1 成立时，执行 Then 后面的语句体 1 中的全部语句，执行完后跳出整个 If 语句体，执行 If 语句体后的语句；当条件不成立时，判断 ElseIf 后的条件 2 是否成立，成立则执行语句体 2 中的全部语句，当条件 2 不成立时，再依次向后判断，如果所有条件都不成立，执行 Else 后面的语句体 n 后跳出整个 If 语句体，执行 If 语句体后的语句。

这种格式的特点是可以进行多个条件的判断，从而执行多个分支，而且 ElseIf 部分可以根据具体情况嵌套多层。

【例 4-4】 同例 4-1，设计一个程序，功能是在窗体的文本框中输入 3 个数，单击“排序”按钮可以对 3 个数进行排序，结果显示在另外一个文本框中。

```
Private Sub Command1_Click()
    a = Val(Text1.Text)
    b = Val(Text2.Text)
    c = Val(Text3.Text)
    If a > b Then
        m1 = a
        m2 = b
    Else
        m1 = b
        m2 = a
    End If
    If c > m1 Then
        Text4.Text = c & "," & m1 & "," & m2
    ElseIf c > m2 Then
        Text4.Text = m1 & "," & c & "," & m2
    Else
        Text4.Text = m1 & "," & m2 & "," & c
    End If
End Sub
```

【例 4-5】 编写一个程序，根据系统时间在窗体上输出“早上好!”、“下午好!”或者“晚上好!”。

代码如下：

```
Private Sub Form_Load()
```

```

Dim h As Integer
Show '使 print 输出在窗体上的内容可见
h = Hour( Time) '取系统的时间
FontSize = 30
If h < 12 Then
    Print "早上好!"
ElseIf h < 18 Then
    Print "下午好!"
Else
    Print "晚上好!"
End If
End Sub

```

4.2.2 Select Case 语句

在有些应用情况下，对某个条件判断后可能会出现多种分支处理的情况，这时虽然 If 语句结构可以进行控制，但是程序显得结构不清晰、代码也不易阅读。这时，如果使用 VB 系统中提供的 Select Case 语句结构，比较理想。

Select Case 语句的一般格式如下：

Select Case 表达式

Case 表达式结果表 1

语句体 1

[Case 表达式结果表 2

语句体 2]

.....

[Case Else

语句体 n]

End Select

功能：根据表达式的值，决定执行几个语句体中的其中之一。

其中，Case Else 部分是可选的，当表达式不匹配 Case 子句的任何部分时执行。如果没有这部分，跳出 Select Case 语句。

Case 表达式结果有几种形式：

(1) 形式 1

Case 表达式结果

这种形式中，“表达式结果表列”中只有一个数值或字符串与表达式的值进行比较。例如 Case 1、Case “a”等。

(2) 形式 2

Case 表达式结果 1[, 表达式结果 2]...[, 表达式结果 n]

这种格式在“表达式结果表列”中有多个数值或字符串供与表达式的值进行比较，这些数值或字符串之间用逗号分隔。如果表达式的值与这多个数值或字符串中的一个相等，即可执行此“表达式结果表列”后相应的语句体部分；否则，若表达式的值与这多个值均不

相等，则需再与其他 Case 后的“表达式结果表列”进行比较。例如 Case 1、3、5、7、9。

(3) 形式 3

Case 表达式结果 1 TO 表达式结果 2

这种形式在“表达式结果表列”中提供了一个数值或字符串的取值范围，可以将此范围的所有取值与表达式的值进行比较。要求表达式结果 1 的值必须小于表达式结果 2 的值，这样才能提供一个合法的范围。如果表达式的值与此范围内的某个值相等，则可执行此“表达式结果表列”后的相应语句体部分；否则，若表达式的值与这个取值范围内的值均不相等，则需要与其后的“表达式结果表列”进行比较。例如 Case 10 TO 30。

(4) 形式 4

Case Is 关系运算符 数值或字符串

这种形式使用了关键字 Is，这里的 Is 表示的是表达式的值。Is 后面只能使用关系运算符 =、<、>、<=、>= 和 <> 等。可以将表达式的值与关系运算符后的数值或字符串进行比较，检验是否满足该关系运算符。若满足，则执行此“表达式结果表列”表后的相应语句体部分；否则，则需再与其后的“表达式结果表列”进行比较。

注意：Select Case 结构只能对一个表达式的结果进行判断，然后再执行不同的操作；而 If 嵌套结构可以对多个表达式的结果进行判断，从而执行不同的操作。

【例 4-6】 设计一个程序，能够判断输入的成绩等级，标准是：100 分等级为“满分”，91~99 分等级为“优秀”，80~89 分等级为“良好”，70~79 分等级为“中等”，60~69 分等级为“及格”，0~59 分等级为“不及格”。

在窗体上放置标签 Label 1、Label 2，以及文本框 Text1 和命令按钮 Command1。判断输入的成绩等级程序控件属性设置见表 4-3。

表 4-3 判断输入的成绩等级程序控件属性设置

控 件	属 性	值	控 件	属 性	值
Label 1	Caption	成绩	Command1	Caption	执行

代码如下：

```
Private Sub command1_click()
```

```
Dim score As Integer, temp As String
```

```
score = Val(Text1.Text)
```

```
temp = "成绩等级为:"
```

```
Select Case score
```

```
Case Is < 60
```

```
Label2.Caption = temp + "不及格"
```

```
Case 60 To 69
```

```
Label2.Caption = temp + "及格"
```

```
Case 70 To 79
```

```
Label2.Caption = temp + "中等"
```

```
Case 80 To 90
```

```
Label2.Caption = temp + "良好"
```

```

Case 99,98,97,96,95,94,93,92,91
    Label2.Caption = temp + "优秀"
Case 100
    Label2.Caption = temp + "满分"
Case Else
    Label2.Caption = "成绩出错"
End Select
End Sub

```

在这个程序使用了 Case 表达式结果表中的 3 种形式,判断输入的成绩等级程序运行后的工作界面如图 4-3 所示。

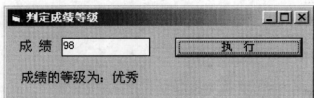


图 4-3 判断输入的成绩等级程序运行后的工作界面

4.2.3 分支嵌套

If 语句和 Select Case 语句都是可以嵌套的,但是不允许出现交叉。如果不交叉,正确的嵌套形式将有很多种,嵌套层次也可以任意多。对于多层 If 嵌套结构中,要特别注意 If 与 Else 的配对关系,一个 Else 必须与 If 配对,配对的原则是,在写含有多层嵌套的程序代码时,建议使用缩进对齐方式,这样容易阅读和维护。

下面是两种正确的嵌套形式。

(1) 形式 1

```

If(条件1)Then
...
If(条件2)Then
...
Else
...
End If
...
Else
...
If(条件3)Then
...
Else
...
End If

```

...

End If

(2) 形式 2

If(条件 1)Then

...

Select Case...

Case...

If(条件 2)Then

...

Else

...

End If

...

Case...

...

End Select

...

End If

4.2.4 GoTo 语句

有时程序遇到不按顺序向下执行,而是直接转向某个指定行的情况时,可以使用 GoTo 语句。使用 GoTo 语句可以改变程序的执行顺序,从当前位置转到指定的位置。

语句格式如下:

GoTo 行标号

参数行标号是必需的,可以是任意的行标签或行号。

行标号是由字母或数字组成,在语句的最前面,且后面要加一个冒号。

【例 4-7】第 2 章的输出对话框程序进行改进。如果没有输入任何值时询问是否继续,如果继续,跳回标号 Line1,如果不继续,跳到标号 Line2。

代码如下:

```
Private Sub Command1_Click()
```

```
Dim MsgBoxValue As Long
```

```
Dim myValue As String
```

```
Line1:
```

```
myValue = InputBox("请输入一个整数", "InputBox 对话框", defaultValue, 100, 100)
```

```
If myValue = "" Then
```

```
MsgBoxValue = MsgBox("没有输入任何值! 是否继续?", vbInformation + vbYesNo,  
"MsgBox 对话框")
```

```
If MsgBoxValue = vbYes Then
```

```
GoTo Line1
```

```
Else
```

```
GoTo Line2
```

```

End If
Else
    If Val(myValue) < 0 Then
        MsgBox "输入的值 < 0!", vbExclamation + vbOKOnly, "MsgBox 对话框"
    Else
        MsgBox "输入值是" + myValue + ", 它的平方根是" + Str(Sqr(Val(myValue)))
            + "!",
            vbInformation + vbOKOnly, "MsgBox 对话框"
    End If
End If
Line2:
End Sub

```

注意：在 VB 系统中，GoTo 语句只能在一个过程内部使用，不能从一个过程用 GoTo 语句转向另一个过程。

正确地使用 GoTo 语句可以大大提高程序的灵活性与简洁性，但太灵活的东西往往是很危险的，它会让程序捉摸不定。结构化程序设计方法，主张限制使用 GoTo 语句，因为滥用 GoTo 语句，将会导致程序结构无规律、可读性差，并且会使程序代码不容易阅读及调试。另外，从功能上说，已经从理论上证明了：任何程序都可以用顺序结构、分支结构和循环结构表示出来，也就是说 GoTo 语句可以取消而用其他方法表达相同的含义。

具体消除 GoTo 语句的方法有：增加辅助变量或者改变程序执行顺序。

4.2.5 支持分支选择的函数和控件

除了上面介绍的语句，VB 系统还有一些函数和控件支持分支结构。

1. IIf 函数

函数格式如下：

IIf(表达式, 值或表达式 1, 值或表达式 2)

3 个参数都是必需的。

功能：根据表达式的值，返回两部分中的其中一个。如表达式为真，返回值或表达式 1；如表达式为假，返回值或表达式 2。

例如，使用 IIf 函数来判断 CheckIt 过程的 score 参数的值，如果参数值大于等于 60，则传回“合格”；否则传回“不合格”。

```

Function CheckIt(score As Integer)
    CheckIt = IIf(score >= 60, "合格", "不合格")
End Function

```

2. Choose 函数

函数格式如下：

Choose(<整数表达式>, <选项 1>[, <选项 2>, ..., <选项 n>])

功能：根据整数表达式的值，从参数列表中选择并返回一个值。

例如，使用 Choose 函数来显示某天的执勤人员情况，strDay 参数传递到过程之中的索引。


```
Function GetChoice(strDay As string)
```

```
strName = Choose(strDay, "张三", "李四", "王五", "赵红", "吴江")
```

```
End Function
```

当 strDay 的值为 1 时, 返回字符串“张三”; 当 strDay 的值为 2 时, 返回字符串“李四”; 以此类推。当 strDay 的值不在 1~5 之间时, 返回 Null。

3. 复选框和单选按钮控件

为了在程序的界面进行分支选择, 可以使用常用控件中的复选框或单选按钮控件。下面通过实例说明两个控件的使用。

【例 4-8】 编写一个程序, 可以改变标签控件的背景颜色和字加粗、加倾斜和加下划线。

在窗体上放一个标签控件 Label1, 3 个复选框和 3 个单选按钮分别放在两个框架中。控件属性的设置见表 4-4。

表 4-4 控件属性的设置

控 件	属 性	值	控 件	属 性	值
Frame1	Caption	背景颜色	Option3	Caption	蓝红色
Frame2	Caption	字体	Check1	Caption	粗体
Option1	Caption	红色	Check2	Caption	斜体
Option2	Caption	绿色	Check3	Caption	下划线

代码如下:

```
Private Sub Check1_Click()
```

```
If Check1.Value = 1 Then
```

```
Label1.FontBold = True
```

```
Else
```

```
Label1.FontBold = False
```

```
End If
```

```
End Sub
```

```
Private Sub Check2_Click()
```

```
If Check2.Value = 1 Then
```

```
Label1.FontItalic = True
```

```
Else
```

```
Label1.FontItalic = False
```

```
End If
```

```
End Sub
```

```
Private Sub Check3_Click()
```

```
If Check3.Value = 1 Then
```

```
Label1.FontUnderline = True
```

```
Else
```

```
Label1.FontUnderline = False
```

```
End If
```

```
End Sub
```

```
Private Sub Option1_Click()  
    If Option1.Value Then Label1.BackColor = &HFF&  
End Sub  
  
Private Sub Option2_Click()  
    If Option2.Value Then Label1.BackColor = &HFF00&  
End Sub  
  
Private Sub Option3_Click()  
    If Option3.Value Then Label1.BackColor = &HFFFFFF00  
End Sub
```

背景颜色和字体设置程序运行后的工作界面如图 4-4 所示。

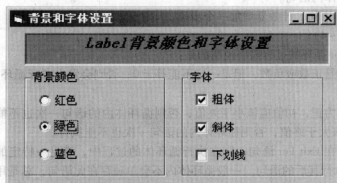


图 4-4 背景颜色和字体设置程序运行后的工作界面

4.3 循环结构程序

在程序设计中，循环结构是指从某处开始有规律的重复执行一段代码的操作，其中，称被重复执行的代码段叫循环体，每一次重复执行都必须判断决定是继续重复还是停止重复执行，做出决定所依据的条件称为循环条件。

用好循环语句可以简化程序代码，让程序变得简单易懂，并且可以提高程序的工作效率。循环结构可以分为 For 循环、While 循环和 Do 循环结构。

4.3.1 For 循环

For 循环以指定次数来重复执行一个循环体。

语句格式如下：

For 循环变量 = 初始值 To 终值 [Step 步长]

[循环体 1]

[Exit For]

[循环体 2]

Next 循环变量

语句说明:

- 1) 循环变量是必要参数, 用作循环计数器的数值变量。
- 2) 初始值和终值都是必要参数, 为循环变量的初值和终值。
- 3) 步长是可选参数。每次循环变量的步长, 如果没有指定, 则 Step 的默认值为 1。
- 4) 循环体 1 和循环体 2 都是可选的。

这些参数中, 要求控制变量是数值型的, 初始值、终值和步长都是数值型表达式。

VB 在执行 For 循环时, 将进行下列操作:

- 1) 设置计数器等于初值。
- 2) 步长大于 0 时, 测试计数器是否大于终值(步长小于 0 时, 测试计数器是否小于终值), 如果是, 退出循环; 如果不是, 进行 3) 操作。
- 3) 执行循环体内语句。
- 4) 执行 Next 语句, 计数器增加一个步长。
- 5) 重复 2) ~ 4)。

VB 在执行 For 循环的过程需要说明的是:

- 1) 步长可以是正数或负数, 但是一定不能等于 0; 否则会发生“死循环”。如果步长不写, 默认为 1。
- 2) 如果步长为正, 初值应该小于终值, 否则循环体内的语句一次也不能执行; 如果步长为负, 初值应该大于终值, 否则循环体内的语句一次也不能执行。
- 3) 若中间存在 Exit For 语句, 则在执行循环体的过程中, 满足一特定的条件即可中间跳出 For 循环, 执行其后的语句。一般循环体内不会单独存在此语句, 总是用一个条件进行控制, 满足时跳出, 不满足继续执行循环体。使用 Exit For 语句的例子见 4.3.5 节。

【例 4-9】 求 $S = 1 \times 2 \times 3 \times \cdots \times n$ 值, 其中 n 由输入对话框获得。

```
Private Sub form_load()  
    Dim s, i As Integer  
    Dim n As Integer  
    n = Val(InputBox("请输入 n 的值:", "输入"))  
    s = 1  
    For i = 1 To n  
        s = s * i  
    Next i  
    MsgBox "s = " & Str(s), 0, "计算结果"  
End Sub
```

除了上述 For 循环形式外, 还有一个集合 For 循环形式。

语法格式如下:

For Each 集合中元素 in 集合

[循环体]

Next [元素]

该语句对集合中的每个元素进行一次循环, 直到集合中无更多的元素时, 执行 Next 后

面的语句。

与 For...Next 循环相比较, For Each...Next 不是将语句运行指定的次数,而是对于数组中的每个元素或对象集合中的每一项重复一组语句。这在不知道集合中元素的数目时非常有用。

当某一数组或集合里的所有元素进行同一操作时,就需要使用该语句了。这里的循环变量必须是变体型或对象型的。对于数组,循环变量必须是变体型的;对于集合,如果集合中包含对象,则循环变量应该是对象型,当然也可以是变体型。

注意: For Each 一般是和集合在一起使用的。

【例 4-10】编写代码列出当前窗体 Form1 上的控件类型。

代码如下:

```
Dim cn As Control
For Each cn In Form1
    Select Case TypeName(cn)
        Case "CommandButton"
            MsgBox "这是一个按钮"
        Case "Label"
            MsgBox "这是一个标签"
        Case "TextBox"
            MsgBox "这是一个文本框"
        Else
            MsgBox "这是其他控件"
        End Select
    Next
```

4.3.2 While 循环

While 循环用于先对条件进行判断,如果条件成立,可以循环执行循环体,直到条件不成立、循环结束为止。与 For 循环最大的差别在于,For 循环用于循环次数已知的情况,执行一定次数后即可结束循环;而 While 循环用于不知道循环次数,但可以用一个条件来进行判断是否结束。

While 循环的格式如下:

```
While 条件
    [循环体]
```

Wend

参数说明:条件是必要参数,可以是数值表达式或字符串表达式,其计算结果为 True 或 False。如果条件为 Null,则条件会视为 False。

该循环格式中的条件一般为条件表达式,结果为布尔类型变量 True 或 False。

功能:只要指定的条件为 True 或非 0,则会重复执行一系列的语句。

VB 在执行 While 循环时,过程如下:

首先判断条件是否成立,若条件成立,则执行循环体内的语句,执行完后再继续判断条件,重复上述过程;否则,结束循环,执行循环后的语句。

While 循环的几点说明:

1) While 循环语句本身不能修改循环条件,所以必须在 While...Wend 语句的循环体内设置相应语句,使得整个循环趋于结束,否则,若初始条件成立,则每次执行完循环体后再检验条件,条件仍然成立,此循环可以无限执行下去,不能结束,变成所谓的“死循环”。

2) While 循环语句先对条件进行判断,然后才决定是否执行循环体。如果开始条件就不成立,则循环体一次也不执行。

3) 凡是用 For...Next 循环编写的程序,都可以用 While...Wend 语句实现。反之,则不然。

【例 4-11】 用 While 循环求 $s = 1 \times 2 \times 3 \times \dots \times n$ 值,其中 n 由输入对话框获得。

```
Private Sub form_load()
    Dim s,i as Integer
    n = InputBox("请输入 n 的值:", "输入")
    s = 1
    i = 1
    While i <= Val(n)
        s = s * i
        i = i + 1
    Wend
    MsgBox "s =" & Str(s), 0, "计算结果"
End Sub
```

【例 4-12】 假设我国现有人口 12 亿,若年增长率为 1.5%,试计算多少年后我国人口增加到或超过 20 亿。

人口计算公式为 $p = y(1+r)^n$ 。其中, y 为人口初值, r 为年增长率, n 为年数。

程序代码如下:

```
Private Sub Form_Click()
    Dim p As Single, r As Single, l As Integer
    p = 12
    r = 0.015
    l = 0
    While p < 20
        p = p * (1 + r)
        l = l + 1
    Wend
    Print l; "年后,我国人口将达到"; p; "亿"
End Sub
```

4.3.3 Do 循环

Do 循环也是根据某个条件是否成立来决定能否执行相应的循环体部分,与 While 循环不同的是: While 循环只能在初始位置检查条件是否成立,若成立,进入循环体;不成立,不进入循环体,执行循环体后的语句。Do 循环可以有两种格式,既可以在初始位置检验条件是否成立,也可以在执行一遍循环体后的结束位置判断条件是否成立,能否进入下一次循环。

Do 循环与前面的 For 循环区别在于,For 循环一般知道循环次数,而 Do 循环一般不知

道循环次数。

Do 循环的两种格式如下:

(1) 格式 1

Do...Loop While | Until 条件

Do

[语句体 1]

[满足某条件时

Exit Do]

[语句体 2]

Loop [While 或 Until 条件]

(2) 格式 2

Do While | Until 条件...Loop

Do [While 或 Until 条件]

[语句体]

[满足某条件时

Exit Do]

[语句体]

Loop

使用说明:

条件是可选参数, 可以是数值表达式或字符串表达式, 其值为 True (非 0) 或 False (0)。如果条件是 Null, 则条件会被当做 False。

1) 格式 1 为先判断后执行, 有可能一次也不执行; 格式 2 为先执行后判断, 至少执行一次循环体。

2) 关键字 Until 用于指明 <条件> 为假 (False) 时, 执行循环体。

3) Exit Do: 当执行该语句时, 退出循环, 执行 While 或 Loop 的下一句。Exit Do 一般放在 If...Then...End If 语句中进行有条件的退出循环。

功能: 当指定的关键字 While 用于指明 <条件> 为真 (True) 或非 0 时, 执行循环体。

上述两种格式构成了 Do 循环的两种使用方法。这两种格式既有共同点又有明显的差异。其共同点为:

1) 均可以有一定的循环条件, 但循环体部分为空: 此种情况下, 可以用于延时, 对于某些程序需要一个短时间的暂停时可以使用这样一个空循环。此方法对于上述 For 循环和 While 循环也同样适用。

2) 均可以从中间用 Exit Do 语句退出循环: 在 Do 循环中可以使用一个或多个 Exit Do 语句, 用于在某些需要中途跳出的情况时使用。可以从所在的 Do 循环中跳出, 执行循环后的语句。

3) 均可以没有循环条件: 此种情况下, 只有 Do...Loop 结构的循环, 整个程序没有结束条件。若循环体为空, 则成为一个死循环; 若在非空循环体中有 Exit Do 语句存在, 也可以在某些条件满足时退出, 否则不能退出, 也成为死循环。

4) 均可以嵌套使用, 而且可以互相嵌套: 两种 Do 循环的嵌套与前面的 For 循环和 While 循环的嵌套类似, 而且这几种循环可以互相嵌套使用。具体嵌套时的执行情况应根据



具体嵌套来决定。

5) 均有两种判断条件的格式: 一种为“While 条件”, 判断时只要条件成立, 则继续执行循环体, 然后重复上述判断过程; 而如果条件不成立, 则退出循环, 执行其后的语句。一种为“Until 条件”, 判断时直到条件成立才退出循环。也即, 条件不成立时, 继续执行循环体, 然后重复判断过程; 当条件成立时, 就要退出循环体, 执行其后的语句。

上述两种格式的不同点为:

1) 第一种格式中的判断条件“While 或 Until 条件”的位置在整个循环体的最后, 而第二种格式中的判断条件“While 或 Until 条件”的位置在整个循环体的起始位置。

2) 第一种格式的执行过程为: 先执行一遍循环体, 在碰到 Loop 后的条件时再进行判断, 根据不同的判断条件格式执行不同的判断过程, 决定是继续执行循环体, 还是退出循环。而第二种格式的执行过程为: 先对条件进行判断, 仍然要根据不同的判断条件格式执行不同的判断过程, 判断为能够执行循环体时, 才能进入循环体内执行相应语句; 否则只能退出循环体。所以, 第一种格式至少要执行一遍循环体, 而第二种格式则可能循环体一遍也不执行。

【例 4-13】编写一个程序, 将十进制整数转换为二进制数, 采用“2 除取余法”, 即将该十进制整数逐次除以 2 而取其余数。

在窗体上放置两个标签控件 Label1、Label2, 两个文本框 TxtDec、TxtBin, 两个命令按钮 CmdBegin、CmdExit。十进制整数转换为二进制数程序控件属性设置见表 4-5。

表 4-5 十进制整数转换为二进制数程序控件属性设置

控 件	属 性	值	控 件	属 性	值
Label1	Caption	“请输入一个十进制整数: ”	TxtBin	Enabled	False
Label2	Caption	“对应的二进制整数为: ”	CmdBegin	Caption	“开始转换”
TxtDec	Text	(空)	CmdExit	Caption	“退出程序”
TxtBin	Text	(空)			

表 4-5 中, 将 TxtBin 控件的 Enabled 属性设置为 False, 是为了该文本框不能编辑。

代码如下:

```
Private Sub CmdBegin_Click()  
    Dim BinStr As String '声明局部变量, 此变量存储转换得的二进制数  
    Dim DecNum As Long  
    Rem 返回字符串内的数字  
    DecNum = Val(TxtDec.Text)  
    BinStr = "" '变量初始化  
    '进行十进制数到二进制数的转换, 直到十进制数等于 0 时结束  
    Do While DecNum <> 0  
        BinStr = (DecNum Mod 2) & BinStr '将十进制数模 2 除得到的结果 1 或 0 连接成字符串  
        DecNum = Int(DecNum/2) '将十进制数除以 2 取整  
    Loop  
    If BinStr = "" Then '判断求得的二进制数是否是空字符串, 如果是则说明转换结果为 0  
        BinStr = "0"  
    End If
```

```

    TxtBin.Text = BinStr '显示得到的二进制数
End Sub

Private Sub CmdExit_Click()
    End
End Sub

```

十进制整数转换为二进制数程序运行后的工作界面如图 4-5 所示。

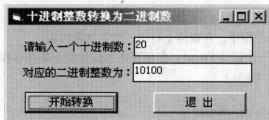


图 4-5 十进制整数转换为二进制数程序运行后的工作界面

4.3.4 循环嵌套

在前面介绍的循环语句中的语句体中可以包含任何合法的 VB 语句，所以也可以包含循环语句。也就是说，在一个循环结构的循环体内含有另一个循环结构，这种结构叫嵌套循环，也叫多重循环。

需要说明的是：

- 1) 此种循环允许嵌套，可以嵌套多层。
- 2) 使用循环嵌套时，内层循环和外层循环的循环控制变量不能相同。
- 3) 外循环必须完全包含内循环，不能交叉。
- 4) 不能从循环体外转向循环体内，也不能从外循环转向内循环。
- 5) 尽量避免太多和太深的循环嵌套结构。
- 6) 循环嵌套结构的书写，最好采用“右缩进”格式，以体现循环层次的关系。

【例 4-14】 设计一个程序，功能是在窗体上打印正三角形的图案，每点界面如图 4-6 所示，单击“显示图形”按钮时，在窗体上显示相应的图形。

代码如下：

```

Private Sub Command1_Click()
    Cls
    For i = 1 To 5
        Print Space(6 - i);
        For j = 1 To i
            Print Space(1); "* ";
        Next j
        Print
    Next i
End Sub

```

运行程序后，单击“显示图形”按钮，得到结果如图 4-6 所示。

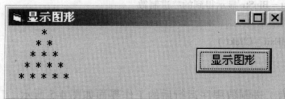


图 4-6 打印正三角形程序运行后的界面

【例 4-15】设计一个程序，在窗体上打印一个九九乘法表。

在新建工程中添加一个窗体 Form1，在该窗体上设计如下事件过程：

```
Private Sub Form_Click()
    FontSize = 12
    Print Tab(35); "九九乘法表"
    Print Tab(33); "-----"
    For i = 1 To 9
        For j = 1 To i
            Print Tab((j - 1) * 9 + 1); i & "*" & j & "=" & i * j;
        Next j
        Print
    Next i
End Sub
```

九九乘法表程序运行后的工作界面如图 4-7 所示。



图 4-7 打印九九乘法表程序运行后的工作界面

4.3.5 中途跳出循环

在 VB 的循环程序中，如果某种情况下需要跳出循环，可以使用以下几种跳出循环语句：

- 1) Exit For: 用于中途跳出 For 循环，可以直接使用，但是通常用条件判断语句加以限制，满足某个条件时，才执行此句，跳出 For 循环。
- 2) Exit Do: 用于中途跳出 Do 循环，同 Exit For 类似，可以直接使用，但是通常用条件判断语句加以限制。

此外，还有 Exit Sub 语句用于中途跳出 Sub 过程，既可以直接使用，也可以用条件判断

语句限制使用; Exit Function 语句用于中途跳出 Function 过程, 可以直接使用, 也可以用条件判断语句限制使用。

【例 4-16】 用 Do 循环设计例 4-11 中的 $s = 1 \times 2 \times 3 \times \cdots \times n$ 的值, 其中 n 由输入对话框获得。

代码如下:

```
Private Sub Command1_Click()  
    n = InputBox("请输入 n 的值:", "输入")  
    s = 1  
    i = 1  
    Do  
        If i > Val(n) Then Exit Do  
        s = s * i  
        i = i + 1  
    Loop  
    MsgBox "s = " & Str(s), 0, "计算结果"  
End Sub
```

【例 4-17】 输入两个正整数, 求它们的最大公约数。

在程序窗体上放置 4 个标签控件 Label1、Label2、Label3、Label4, 2 个文本框 Text1、Text2, 1 个命令按钮 Command1。最大公约数程序控件属性设置如表 4-6 所示。

表 4-6 最大公约数程序控件属性设置

控 件	属 性	值	控 件	属 性	值
Label1	Caption	输入两个整数	Label4	Caption	最大公约数
Label2	Caption	m =	Command1	Caption	计算
Label3	Caption	n =			

代码如下:

```
Private Sub Command1_Click()  
    Dim m As Integer, n As Integer, p As Integer, y As Integer  
    Do  
        m = Val (Text1.Text)  
        n = Val (Text2.Text)  
        If m <= 0 Or n <= 0 Then  
            y = MsgBox ("数据错误! 是否继续?", vbInformation + vbYesNo)  
            If y < > vbYes Then  
                Exit Sub  
            End If  
        End If  
        Exit Do  
    End If  
    Loop  
    If m < n Then
```

Else

```

x = m
m = n
n = x
End If
For i = 1 To n
    p = m Mod n
    m = n
    n = p
    If p = 0 Then Exit For
Next
Text3.Text = m
End Sub

```

最大公约数程序运行后的工作界面如图 4-8 所示。

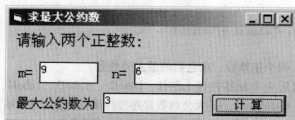


图 4-8 最大公约数程序运行后的界面

4.3.6 循环语句的小结

1) 在各种循环结构中, 要注意正确设置循环条件, 因为如果永远不能满足退出循环的条件, 将会导致出现不断执行循环体的现象, 严重的会导致死机。程序设计中应避免这种“死循环”。

例如, 下面两段代码都导致了死循环:

```

Dim i As Integer
For i = 1 To 10
    Beep
    Debug.Print i
    If i = 9 Then i = 1
Next
Dim i As Integer
i = 10
Do
    Beep
Loop While i >= 8

```

2) 使循环体内工作量最小化来提高程序的时间效率。

3) 在多重嵌套循环中, 如有可能, 应当将最长的循环放在内层, 最短循环放在外层, 这样就可以减少 CPU 跨切循环层的次数, 从而优化程序性能。

4) 尽量减少循环嵌套层次。

【例 4-18】 下面分别使用 For-Next、DoWhile-Loop、Do-LoopWhile、DoUntil-Loop、Do-LoopUntil、Do-Loop 方法，计算从 1 加到 100 的和，结果显示在一个文本框中。从 1 加到 100 的和程序控件属性设置见表 4-7。

表 4-7 从 1 加到 100 的和程序控件属性设置

控 件	属 性	值	控 件	属 性	值
Text1	Text	求和结果	Command4	Caption	DoUntil-Loop
Command1	Caption	For-Next	Command5	Caption	Do-LoopUntil
Command2	Caption	DoWhile-Loop	Command6	Caption	Do-Loop
Command3	Caption	Do-LoopWhile			

代码如下：

```
Private Sub Command1_Click()
```

```
    Show
```

```
    s = 0
```

```
    For k = 1 To 100
```

```
        s = s + k
```

```
    Next k
```

```
    Text1.Text = "For-Next 方法:s=" + CStr(s)
```

```
End Sub
```

```
Private Sub Command2_Click()
```

```
    n = 0
```

```
    s = 0
```

```
    Do While n < 100
```

```
        n = n + 1
```

```
        s = s + n
```

```
    Loop
```

```
    Text1.Text = "Do While-Loop 方法:s=" + CStr(s)
```

```
End Sub
```

```
Private Sub Command3_Click()
```

```
    n = 0
```

```
    s = 0
```

```
    Do
```

```
        n = n + 1
```

```
        s = s + n
```

```
    Loop While n < 100
```

```
    Text1.Text = "Do-Loop While 方法:s=" + CStr(s)
```

```
End Sub
```

```
Private Sub Command4_Click()
```

```
n = 0
```

```
s = 0
```

```
Do Until n >= 100
```

```
    n = n + 1
```

```
    s = s + n
```

```
Loop
```

```
Text1.Text = "DoUntil-Loop 方法:s =" + CStr(s)
```

```
End Sub
```

```
Private Sub Command5_Click()
```

```
    n = 0
```

```
    s = 0
```

```
    Do
```

```
        n = n + 1
```

```
        s = s + n
```

```
    Loop Until n >= 100
```

```
    Text1.Text = "Do-LoopUntil 方法:s =" + CStr(s)
```

```
End Sub
```

```
Private Sub Command6_Click()
```

```
    n = 0
```

```
    s = 0
```

```
    Do
```

```
        n = n + 1
```

```
        s = s + n
```

```
    If n >= 100 Then Exit Do
```

```
    Loop
```

```
    Text1.Text = "Do-Loop 方法:s =" + CStr(s)
```

```
End Sub
```

从 1 加到 100 的和程序运行后的工作界面如图 4.9 所示。

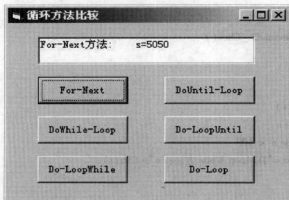


图 4-9 从 1 加到 100 的和程序运行后的工作界面

习 题

4-1 设计一个窗体,输入一个3位整数,将它反向输出。例如输入123,输出为321。

4-2 设计一个窗体,输入3个整数,显示其中的最小值。

4-3 编写一个程序,打印如图4-10所示的杨辉三角形。

4-4 设 $s = 1 \times 2 \times 3 \times \cdots \times n$, 求 s 不大于5000时最大的 n 。

4-5 由键盘输入 N 个数,分别统计其中正数的个数、负数的个数和零的个数。



图4-10 习题4-3图

4-6 编写一个程序,输入球的半径,计算球的体积和表面积,在输出对话框中显示。

4-7 编写一个程序,输出1~100之间所有不能被7整除,也不能被3整除的整数。

4-8 编写一个程序,选择1~12月中的某月,在文本框中显示该月对应的季节。例如选择2月,输出:2月是冬季。

4-9 编写一个程序,求 $S = 1 - 1/2 + 1/3 - 1/4 + \cdots + 1/99 - 1/100$ 的值。

4-10 在窗体上输出图4-11所示图形。

4-11 用 `if` 函数求分段函数: $y = \begin{cases} 2x+3 & (x < 0) \\ 3-2x & (x \geq 0) \end{cases}$ 的值。

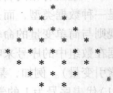


图4-11 习题4-10图

4-12 编写一个程序,输入一个年份,输出该年份是否是闰年。闰年的条件是:年份能被4整除,但不能被100整除;或者能被400整除。

4-13 输入3个数,将它们从小到大排序。

4-14 编写一个“计算三角形的面积”程序,给定三角形的3条边长,判断给出的3条边能否构成三角形,如果能构成,计算三角形的面积,如果不能构成,提示要求重新输入。

4-15 编写一个程序,运行时鼠标单击窗体,按一次在窗体上显示“春晓”诗的一句,诗显示完整后,将窗体上的诗清除后再重新开始显示。

“春晓”

“春眠不觉晓”

“处处闻啼鸟”

“夜来风雨声”

“花落知多少”

4-16 求一元二次方程 $ax^2 + bx + c = 0$ 的解,若 $a = 0$,提示“不是二次方程”;若 $b^2 - 4ac = 0$,提示“有两个相等实根”;若 $b^2 - 4ac > 0$,提示“有两个不等实根”;若 $b^2 - 4ac < 0$,提示“有两个共轭复根”。

4-17 求 $S = 1^2 + 2^2 + \cdots + 100^2$ 的值。

4-18 编写一个程序,可以将十进制整数转化为八进制数。

4-19 输入一个正整数,将该数写成若干个素数相乘的形式,例如, $10 = 2 \times 5$, $24 = 2 \times 2 \times 2 \times 3$, $76 = 2 \times 2 \times 19$,结果在输出对话框显示。

4-20 输入任意一个数,求函数 $y = \begin{cases} x^2 & (x > 0) \\ 0 & (x = 0) \\ x & (x < 0) \end{cases}$ 的值。

4-21 输入任意一个正整数,判断该数是否是素数。

4-22 编写一个程序,功能是输入一个大于等于6的偶数,将它表示成为两个素数之和,将结果打印出来,从而验证哥德巴赫猜想。

第5章 数 组

前面各章节所介绍的例子,使用的都是属于简单数据类型(整型、字符串、浮点型)的数据,可以通过简单变量来访问它的元素,如 N、I1、I5 等。然而,在实际应用中经常要处理性质相同的成批数据,例如矩阵运算、输出表格、数据排序等,若按简单变量的使用规则,要逐一命名、逐个运算、逐个输出,使程序的书写变得十分烦琐冗长。在这种情况下,应采用专门用来处理这类群体性数据的方法——数组。

5.1 数组概述

一组具有相同名字、不同下标的用来表示一组具有相同性质的数据称为数组。数组并不是一种数据类型,而是一组相同类型的变量的集合,该集合的名字即数组名。数组名的命名规则与简单变量的命名规则相同。数组中的每一个数据称为一个数组元素,用数组名和该数据在数组中的序号来唯一标识,序号即下标(或索引)。因此,数组元素也称为下标变量(或索引变量)。例如,某班有 50 名学生,可以用一个数组 E 来表示 50 个学生的英语成绩, E(1)代表序号为 1 的学生的英语成绩, E(2)代表序号为 2 的学生的英语成绩, ..., E(50)代表序号为 50 的学生的英语成绩。例如:

E(1)、E(2)、E(3)、...、E(50)

其中, E 是数组名,括号内是数组元素的下标,而 E(1)、E(2)、...、E(50)等是该数组的数组元素(下标变量)。

使用数组可以方便灵活地组织和使用的数据。数组元素在程序中的使用与简单变量类似,在简单变量允许出现的多数地方也允许出现数组元素。例如, A = 98 是给简单变量 A 赋值,同样也可以通过 E(2) = 100 给数组元素 E(2)赋值。

在表示数组元素时,应注意以下几点:

1) 要用圆括号把下标括起来,不能用中括号或大括号代替,也不能省略圆括号。例如,数组元素 E(9)表示成 E[9]、E{9}或 E9 都是错误的。

2) 下标的最小取值称为下界,下标的最大取值称为上界。在不加任何说明的情况下,数组下标的下界默认值为 0。

3) 下标可以是常量、变量或表达式,其值必须是整数,否则将被自动四舍五入为整数;此外,下标值必须在所定义的上、下界范围之内,否则发生越界错误。

对于一个数组,如果只用一个下标就能确定一个数组元素在数组中的位置,则称为一维数组,也就是说,由一个下标的数组元素所组成的数组称为一维数组,上面的数组 E 即是一维数组。而由具有两个或多个下标的数组元素所组成的数组称为二维数组或多维数组。为了描述一个数组的大小(即指定各维的下标界限),就需要用语句定义。下面是二维数组 M(2,3)包含的数组元素:

M(0,0) M(0,1) M(0,2) M(0,3)

M(1,0) M(1,1) M(1,2) M(1,3)

M(2,0) M(2,1) M(2,2) M(2,3)

根据数组占用存储空间的方式不同, Visual Basic 中的数组分为两种: 固定大小的数组和动态数组。在定义数组时已经确定了数组的大小, 称为固定大小的数组; 在定义数组时未给出数组的大小, 称为动态数组。固定大小的数组对应的内存是在编译过程中分配的, 即程序尚未运行, 数组占用的内存大小便已确定, 程序运行时其大小不能再改变; 而动态数组对应的内存是在程序运行时分配的, 并且运行时其大小可以根据需要随时调整。这两种数组的定义方法不同, 使用方法也略有不同。

5.2 数组声明

数组必须先声明后使用。本小节讨论一维数组的声明方式, 包括一维固定大小数组和一维动态数组的声明方式, 多维数组的声明方式见 5.4 小节。

5.2.1 固定大小的数组的定义

1. 格式

格式如下:

<说明符> <数组名> (<界标>)[As <数据类型>]

2. 功能

功能如下:

声明一个一维数组, 包括确定一维数组的名称、数组元素的个数和类型, 并为一维数组分配存储空间。

3. 说明

1) 格式中的说明符为 Visual Basic 的保留字, 不同的说明符可以定义数组的不同性质和作用范围, 其定义形式如下:

Dim | Static | Private | Public

其中:

Dim: 可以用于模块、窗体和过程。

Static: 只能用于过程和函数体内。经过过程和函数的调用, 数组元素仍能保持原有值。

Private: 可以用于模块和窗体, 不能用于过程和函数体内。

Public: 仅能用于“.bas”模块, 不能用于窗体、过程和函数体内。

数组的作用域如下:

用 Public 声明的数组作用域为整个程序。

用 Private 声明的数组作用域分模块级和窗体级。

用 Dim 声明的数组作用域分模块级、窗体级和过程级。

用 Static 声明的数组作用域为过程级, 但数组元素经过过程后不再是初始化的值。

2) 格式中的数组名与简单变量的命名规则相同。

3) 格式中的界标定义形式如下:

[<下界> To <上界>]

下界和上界规定了一维数组元素下标的取值范围。省略下界时, Visual Basic 默认其值为 0, 可以使用 Option Base 语句将默认下界修改为 1, 其格式如下:

Option Base |0|1|

注意, Option Base 语句只能在模块级中使用, 用来声明数组下标的默认下界, 它只影响位于包含该语句的模块中的数组的下界。使用时必须把它放在数组定义之前。

一维数组所含元素个数为:

上界 - 下界 + 1

4) 格式中的数据类型的可以是 Integer、Long、Single、Double、Boolean、String(可变长度字符串)、String * n(固定长度字符串)、Currency、Byte、Date、Object、Variant、用户定义类型或对象类型。与声明变量类似, 一个 As <数据类型> 只能定义一个数组的类型。

5) 数组必须先定义, 后使用。声明数组后, VB 自动对数组元素进行初始化, 将数值型数组元素值置为 0, 将字符串型数组元素值置为零长度字符串, 将可变类型数组元素值置为 Empty。

6) 在编译时计算机为固定大小的数组分配固定的存储空间, 在运行期间大小不能改变。

例如:

```
Dim A(5) As Integer
```

声明了一个具有 6 个元素的一维整型数组, 其下界默认为 0, 上界为 5, 包含的数组元素有 A(0)、A(1)、A(2)、A(3)、A(4)、A(5)。

例如:

```
Dim B(-3 To 3) As Integer
```

声明了一个具有 7 个元素的一维整型数组, 其下界为 -3, 上界为 3, 包含的数组元素有 B(-3)、B(-2)、B(-1)、B(0)、B(1)、B(2)、B(3)。

对于固定大小的数组, 由于在程序运行时其所占内存大小不能改变, 因此在使用时, 要按照所处理数据可能达到的最大个数为其声明数组空间, 这样会浪费一定的内存空间。程序中过度使用固定大小的数组会对整个系统的性能造成影响。

5.2.2 动态数组的定义

在解决实际问题时, 经常遇到不能提前得知所需数组到底应该有多大才合适的情况, 所以希望能够在运行时改变数组的大小。使用动态数组就可以在任何时候改变其大小, 并且可以在暂时不需要时清除动态数组所占用的存储空间, 而后再需要时, 重新申请新的存储空间。例如, 可以在短时间内使用一个大数组, 然后, 在不使用这个数组时, 将内存空间释放掉, 下次又需要使用大数组则再次要系统为其分配存储空间。因此, 使用动态数组有助于高效使用内存, 提高系统的性能。

1. 格式

定义动态数组需要分以下两步进行:

1) 在模块级或过程级按以下格式定义一个没有界标的数组:

<说明符> <数组名>() [As <数据类型>]

这里的说明符、数组名和数据类型的要求同固定大小的数组的定义相同。

2) 在过程级使用下面的 ReDim 语句分配数组的实际元素个数:

ReDim <数组名>(<维数定义>)[As <数据类型>]

这里的 <维数定义>通常包含变量或表达式, 但其中的变量或表达式应有明确的值。

2. 说明

1) 在定义动态数组的两个步骤中, 如果步骤 1) 已经定义了数组的类型, 则不允许用步骤 2) 改变类型定义。

2) ReDim 语句只跑出现在过程中, 与 Dim 语句不同, ReDim 语句是一个可执行语句。

3) 可以用 ReDim 语句反复改变数组元素及维数的数目。

4) 每次执行 ReDim 语句时, 当前存储在数组中的值会全部丢失。VB 重新对数组元素进行初始化, 即将数值型数组元素值置为 0, 将字符串型数组元素值置为零长度字符串, 将可变类型数组元素值置为 Empty。

例如:

```
Option Base 1

Dim Z() As Integer

Private Sub Command1_Click()

    M = 10

    ReDim Z(M)

    For I = 1 To M
        Z(I) = I
    Next I
    Print Z(I)

End Sub

Private Sub Command2_Click()

    L = 20

    ReDim Z(L)

    For I = 1 To L
        Z(I) = I + 1
    Next I
    Print Z(I)

End Sub
```

以上程序在窗体模块的声明段使用 Dim Z() As Integer 语句将 Z 定义为一个动态整型数组, 在命令按钮 Command1 的 Click 事件过程中, 使用 ReDim Z(M) 将 Z 定义为一个具有 10 个元素的一维数组, 在命令按钮 Command2 的 Click 事件过程中, 使用 ReDim Z(L) 将 Z 定义为一个具有 20 个元素的一维数组。

5.3 数组操作

数组在声明之后就可以使用了, 即可以对数组元素进行各种操作。数组的常用操作有数组的初始化, 数组元素的输入、输出、复制等。在很多情况下, 将数组元素的下标和循环语

句结合使用,能解决大量的实际问题。

5.3.1 数组元素的引用

访问数组内存放的数据(即数组元素)的格式如下:

数组名(下标)

数组元素可以参加其类型所允许的各种操作。需要指出的是,数组定义时用数组名表示数组的整体,具体操作时是针对每个数组元素进行的。

例如:

```
Dim A(10) As Integer
```

```
...
```

```
A(5) = A(1) + 10
```

```
...
```

以上程序中定义了一个整型数组 A,因为它是数值型数组,因此后面 A(5) = A(1) + 10 赋值语句中进行的算术运算操作成立。

5.3.2 给数组元素赋初值

1. 利用 Array 函数

在 VB 中可以使用 Array 函数为数组元素赋值,即把一个数据集赋值给某个数组。

格式如下:

<数组变量名> = Array(数组元素值)

功能:将数组元素值赋给数组各元素的值,各值之间用逗号分隔。

说明:数组变量名是预先定义的数组名(定义时没有指定维数和上、下界,并且类型必须为变量体类型 Variant),在数组变量名之后没有括号。

例如:

```
Dim Week(), Num
```

```
Week = Array("SUN", "MON", "TUE", "WED", "THU", "FRI", "SAT")
```

```
Num = Array(0, 1, 2, 3, 4, 5)
```

```
Print Week(0)
```

```
Print Num(0)
```

以上程序先定义数组变量名 Week 和 Num,然后用函数 Array 为它们赋初值,最后输出每个数组中的第一个元素的内容,输出结果如下:

```
SUN
```

```
0
```

注意:

1) Array 函数只能对一维数组各元素赋值,定义数组时圆括号可以省略,其类型只能是 Variant。

2) 数组的下界为零,上界由 Array 函数括号内的参数个数决定,也可通过函数 UBound 获得。

2. 利用 For 循环语句

例如：

```
Dim T(10) As Integer
For I = 0 To 10
    T(I) = 1
Next I
```

以上程序先定义数组 T，然后通过 For 循环语句为数组中的每一个元素赋初值为 1。

3. 利用赋值语句

如果数组较小时，可以用赋值语句来实现数组元素赋初值。

例如：

```
Dim B(1 To 3) As Integer
B(1) = 1
B(2) = 2
B(3) = 3
```

以上程序先定义数组 B，然后通过 3 个赋值语句为数组的 3 个元素赋初值。

5.3.3 数组的输入

数组的输入可以用 For 循环语句及文本框控件输入，也可以通过 InputBox 函数输入。

例如：

```
Dim N(1 To 3) As String
For I = 1 To 3
    N(I) = InputBox("请输入名字")
Next I
```

以上程序定义数组 N 为字符串型数组，包含 3 个数组元素，然后在 For 循环语句中利用 InputBox 函数为数组中的每一个元素实现数据的输入。

对于数组元素的输入，既可以对整个数组输入数据，也可以对个别数组元素输入数据。

例如：

```
Dim A(1 To 10) As Integer
A(2) = Val(Text1.Text)
A(5) = Val(Text2.Text)
```

以上程序定义数组 A，并默认将数组 A 的各元素初始化为 0，然后通过文本框 Text1 和 Text2 为数组中的第 2 个元素和第 5 个元素输入数据。注意：由于数组 A 是数值类型，而文本框 Text 属性返回的是字符串类型的数据，故需用函数 Val() 实现数据类型的转换。

5.3.4 数组的输出

数组元素经处理后，常需要将处理结果显示给用户，这就涉及到数组元素的输出。可以使用 Print 方法输出数组元素，也可以使用控件输出数组元素，如标签、文本框、组合框、列表框等。

例如：

```
Dim H(1 To 5) As String
H(1) = "one"
```



```
H(2) = "two"  
H(3) = "three"  
H(4) = "four"  
H(5) = "five"  
For I = 1 To 5  
    Print H(I)  
Next I
```

以上程序定义数组 H 为字符串型数组，然后通过赋值语句为数组的各个元素赋初值，最后通过一个 For 循环语句遍历数组的各个元素，并用 Print 语句将其一一输出。

又如：

```
Dim H(1 To 5) As String  
For I = 1 To 5  
    H(I) = InputBox("请输入单词")  
Next I  
Text1.Text = ""  
For I = 1 To 5  
    Text1.Text = Text1.Text & H(I) & Space(1)  
Next I
```

以上程序定义数组 H 为字符串型数组，然后在第一个 For 循环语句中利用 InputBox 函数为数组中的每一个元素实现数据的输入，最后在第二个 For 循环语句中利用文本框 Text1 将数组元素输出。注意：使用文本框显示多个数据时，常需要给文本框设置 MultiLine 属性为 True，并设置滚动条，依据滚动条的方向，要注意每显示一个或多个数组元素后是否要在文本框中加上回车或换行符号。

再如：

```
Dim H  
H = Array("one", "two", "three", "four", "five")  
Label1.Caption = ""  
For I = 0 To 4  
    Label1.Caption = Label1.Caption & H(I) & Space(5)  
Next I
```

以上程序先定义数组变量名 H，然后用函数 Array 为其赋初值，最后在 For 循环语句中利用标签控件 Label1 将数组元素输出。注意：使用标签控件显示多个数据时，应注意设置标签的 AutoSize 属性为 True，如果希望标签大小能随文本内容自动垂直扩展，还应将其 WordWrap 属性设置为 True。

5.3.5 数组元素的复制

数组元素的复制就是将一个数组中的元素复制到另一个数组的元素中。单个数组元素的复制可以像简单变量一样用赋值语句实现，整个数组的复制需要用循环来实现。

例如：

```
Dim A(5) As Integer, B(10) As Integer  
...
```

```
A(1) = B(1)
```

```
A(2) = B(2)
```

```
...
```

又如:

```
Option Base 1
```

```
...
```

```
Dim A(15) As Integer, B(15) As Integer
```

```
...
```

```
For I = 1 To 15
```

```
    A(I) = B(I)
```

```
Next I
```

再如:

```
Option Base 1
```

```
...
```

```
Dim A(10) As Integer, B() As Integer
```

```
...
```

```
ReDim B(10)
```

```
For I = 1 To 10
```

```
    B(I) = A(I)
```

```
Next I
```

```
...
```

5.3.6 数组的清除

在一个程序中,同一数组只能用 Dim 语句定义一次。但有时可能需要清除数组的内容或对数组重新定义,这可以用 Erase 语句来实现。

格式如下:

```
Erase <数组名> [, <数组名>]
```

功能:用于重新初始化固定大小数组的元素,或者释放动态数组的存储空间。

说明:

- 1) 在 Erase 语句中,只需给出数组名,不带括号和下标。
- 2) 在 Erase 语句用于固定大小数组时,如果这个数组是数值数组,则把数组中的所有数组元素置为 0;如果是字符串数组,则把所有数组元素置为空字符串;如果是可变类型数组,则把数组中的所有数组元素置为 Empty。注意, Erase 语句不能释放固定大小数组所占的存储空间。
- 3) 当把 Erase 语句用于动态数组时,将删除整个数组结构并释放该数组所占用的内存空间。也就是说,动态数组经 Erase 语句执行后,即不复存在;而固定大小数组经 Erase 语句执行后,仍然存在,只是其内容被清空。
- 4) Erase 语句释放动态数组所使用的内存,在下次引用该动态数组之前,必须用 ReDim 语句,重新定义该数组。

例如:



```

Dim A(1 To 5) As Integer, B(1 To 5) As Integer
For I = 1 To 5
    A(I) = I
    B(I) = I + 5
Next
For I = 1 To 5
    Print A(I);
Next
For I = 1 To 5
    Print B(I);
Next
Print
Erase A, B
For I = 1 To 5
    Print A(I);
Next
For I = 1 To 5
    Print B(I);
Next

```

以上程序定义了两个固定大小数组 A、B，并用 For 循环语句为每个数组的元素赋值，然后执行 Erase 语句，将各数组元素的值清除。程序的运行结果如下：

1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0

5.3.7 针对数组的循环语句 For Each...Next

For Each...Next 语句类似于 For...Next 语句，两者都用来执行指定重复次数的一组操作。而 For Each...Next 是专门用于数组或对象的。

格式如下：

For Each <变量> In <数组名>

[<语句组 1>]

[Exit For]

[<语句组 2>]

Next <变量>

功能：用于对数组中的所有元素重复进行类似的操作。

说明：

1) 格式中的变量只能是一个可变类型的变量，它是为循环提供的，代表数组中的每个元素。

2) 不能在 For Each...Next 语句中使用用户自定义类型的数组，因为 Variant 不能包含用户定义类型。

3) 不含 Exit For 语句时，语句组 1 和语句组 2 重复的次数由数组中元素的个数确定，有多少个元素，就会重复执行多少次。如果包含 Exit For 语句，则该语句通常应包含在条件

语句中,当条件满足时结束循环操作。

5.4 多维数组

前面各小节主要以一维数组为例,介绍了数组的使用方法。除了一维数组,在解决诸如矩阵运算、行列式运算时,经常需要使用二维数组。根据实际问题的需要,也可能选择使用三维数组、四维数组等更高维数的数组,VB 中允许数组的最大维数是 60。

5.4.1 多维数组的声明

1. 格式

格式如下:

<说明符> <数组名> (<维数定义>)[As <数据类型>]

2. 功能

功能如下:

声明一个多维数组,包括确定多维数组的名称、维数、每一维的大小和数组元素的类型,并为多维数组分配存储空间。

3. 说明

1) 格式中的<维数定义>形式为:

[<下界 1> To <上界 1>[, <下界 2> To <上界 2>[, ...]]

其中,下界和上界规定了多维数组元素每一维下标的取值范围。

2) 多维数组中数组元素的个数等于每一维的大小之积,即 n 维数组元素个数为

$(\text{上界 } 1 - \text{下界 } 1 + 1) \times (\text{上界 } 2 - \text{下界 } 2 + 1) \times \cdots \times (\text{上界 } n - \text{下界 } n + 1)$

3) 格式中的<说明符>、<数组名>、<数据类型>等内容的要求,同一维数组声明中的要求相同。

多维数组中数组元素的引用形式为:

数组名(下标 1, 下标 2, 下标 3, ...)

例如:

```
Dim E(1 To 3, 0 To 5) As Long
```

声明了一个具有 3×6 个元素的二维长整型数组,其第一维下界为 1,上界为 3,第二维下界为 0,上界为 5,第二维下界 0 可以省略,即写成 `Dim E(1 To 3, 5) As Long`。

该二维数组包含的数组元素如下:

E(1,0) E(1,1) E(1,2) E(1,3) E(1,4) E(1,5)

E(2,0) E(2,1) E(2,2) E(2,3) E(2,4) E(2,5)

E(3,0) E(3,1) E(3,2) E(3,3) E(3,4) E(3,5)

5.4.2 多维数组的操作

前面以一维数组为例介绍的数组的输入、输出、复制、清除等操作同样适用于多维数组。在对多维数组进行操作时,应了解多维数组的内存存放形式,以决定遍历多维数组元素的次序。

任何维数的多维数组在内存中都是以一维线性的形式存放的,多维数组逻辑上的多维与实际物理存储的一维之间存在一种对应关系。这种对应关系决定了多维数组各个元素在内存中存放的先后顺序,对于多维数组元素的访问,应按照这个先后顺序用嵌套的循环语句进行遍历。

以二维数组为例,其在内存中的存放形式是以行优先方式存放。即先存放每一行中的各个列元素,一行存储完毕,再存储下一行。

例如,二维数组 A(2,3)包含如下数组元素:

A(0,0) A(0,1) A(0,2) A(0,3)

A(1,0) A(1,1) A(1,2) A(1,3)

A(2,0) A(2,1) A(2,2) A(2,3)

该数组在内存中的存放顺序为 A(0,0)、A(0,1)、A(0,2)、A(0,3)、A(1,0)、A(1,1)、A(1,2)、A(1,3)、A(2,0)、A(2,1)、A(2,2)、A(2,3)。

因此,二维数组 A(2,3)应按如下方法遍历:

```
For i = 0 To 2
```

```
For j = 0 To 3
```

```
...
```

```
...A(i,j)...
```

```
...
```

```
Next j,i
```

行优先顺序推广到维数更多的多维数组,可规定为先存储最右下标元素,依次向左完成各维的存储。

例如:三维数组 B(1,2,3)的内存存放顺序为 B(0)(0)(0)、B(0)(0)(1)、B(0)(0)(2)、B(0)(0)(3)、B(0)(1)(0)、B(0)(1)(1)、B(0)(1)(2)、B(0)(1)(3)、B(0)(2)(0)、B(0)(2)(1)、B(0)(2)(2)、B(0)(2)(3)、B(1)(0)(0)、B(1)(0)(1)、B(1)(0)(2)、B(1)(0)(3)、B(1)(1)(0)、B(1)(1)(1)、B(1)(1)(2)、B(1)(1)(3)、B(1)(2)(0)、B(1)(2)(1)、B(1)(2)(2)、B(1)(2)(3)。

因此,三维数组 B(1,2,3)应按如下方法遍历:

```
For i = 0 To 1
```

```
For j = 0 To 2
```

```
For k = 0 To 3
```

```
...
```

```
...B(i)(j)(k)...
```

```
...
```

```
Next k,j,i
```

对于 n 维数组 Z(下界 1 To 上界 1,下界 2 To 上界 2,...,下界 n To 上界 n),其遍历顺序如下:

```
For i = 下界 1 To 上界 1
```

```
For j = 下界 2 To 上界 2
```

```
For k = 下界 3 To 上界 3
```

```
...
```

For m = 下界 n To 上界 n

...

...Z(i)(j)(k)...(n)

...

Next m,...,k,j,i

在访问多维数组元素时，应尽量按照以上介绍的顺序遍历。当然，不按照这个顺序同样可以遍历数组元素，但程序的执行效率会受影响，建议不要这样做。

动态数组既可以被重新定义为一维数组，也可以被重新定义为多维数组。

Option Base 1

Private Sub Command1_Click()

Dim B() As Integer

ReDim B(5)

For i = 1 To 5

B(i) = i

Print B(i)

Next i

ReDim B(2,3)

For i = 1 To 2

For j = 1 To 3

B(i,j) = (i-1)*3 + (j-1)

Print B(i,j)

Next j,i

End Sub

以上程序在 Command1_Click 子过程中首先使用 Dim B() As Integer 语句将 B 定义为一个动态整型数组，接着使用 ReDim B(5) 将 B 定义为一个具有 5 个元素的一维数组，对其进行赋值、输出操作，然后使用 ReDim B(2,3) 将 B 定义为一个具有 6 个元素的二维数组。

5.4.3 保留动态数组的内容

动态数组可以使用 ReDim 语句进行多次定义，但每次使用 ReDim 语句进行定义时，数组就会被重新初始化，存储在数组中的值就会全部丢失。如果在改变数组大小时，不希望数组中原有的数据丢失，可以使用带 Preserve 参数的 ReDim 语句。

格式如下：

ReDim Preserve <数组名> (<维数定义>)

功能：用于重新定义数组的大小，同时保留原数组中元素的值。

说明：用 Preserve 参数只能改变多维数组中最后一维的上界，如果改变了其他维或最后一维的下界，运行时就会出现错误。

例如：

Option Base 1

Dim Z() As Integer

...

ReDim Z(5)

```
For I = 1 To 5
```

```
    Z(I) = I
```

```
Next
```

```
ReDim Preserve Z(10)
```

```
For I = 1 To 10
```

```
    Print Z(I);
```

```
Next
```

以上程序定义了一个整型动态数组 Z，并用 ReDim 语句对其进行了两次重定义。第一次重定义，将其数组长度定义为 5，并用一个循环语句为其 5 个数组元素依次赋值。第二次重定义，将其数组长度定义为 10，由于带有 Preserve 参数，故保留了前 5 个元素原有的值，程序的输出如下：

```
1    2    3    4    5    0    0    0    0    0
```

5.4.4 数组标界的检测函数

程序有时需要知道数组的上界值和下界值，这可以通过 UBound 和 LBound 函数来检测。

格式如下：

UBound(数组名[, 维])

LBound(数组名[, 维])

功能：UBound 函数返回一个数组中指定维的上界；LBound 函数返回一个数组中指定维的下界。

说明：

1) 格式中的[, 维]是用于指定要测试的数组的某一维。两个函数一起使用，即可确定一个数组的大小。

2) 对于一维数组来说，[, 维]可以省略。如果要测试多维数组，[, 维]不能省略。

例如：

```
Dim A(1 To 10, 0 To 5, 2 To 8)
```

```
Print LBound(A, 1), UBound(A, 1)
```

```
Print LBound(A, 2), UBound(A, 2)
```

```
Print LBound(A, 3), UBound(A, 3)
```

以上程序测试三维数组每维的下界和上界的值。

输出结果如下：

```
1    10
```

```
0    5
```

```
2    8
```

5.5 控件数组

在程序设计中，常常要使用一些类型相同、功能相似的控件。为了便于管理、提高效率，VB 中提供了控件数组的机制，它为处理一组功能相近的控件提供了有力方法。

5.5.1 控件数组的概念

控件数组是由一组相同类型的控件组成。例如,可以将一批功能类似的命令按钮定义成一个控件数组,也可以将一批功能类似的文本框定义成一个控件数组。与前面介绍的数组变量相同,控件数组中的每一个控件是该控件数组的一个元素,表示如下:

<控件数组名>(<索引>)

同一控件数组名称相同,具有相同的属性,但各数组元素通常有不同的属性值(所有元素的 Name 属性必须相同)。当控件数组建立时,系统给每个元素赋一个唯一的索引号(Index),即下标。下标值由控件的 Index 属性指定,从 0 开始(即第 1 个元素下标为 0,第 2 个元素下标为 1,以此类推),最大为 32767。通过控件属性窗口的 Index 属性,可以知道该控件的下标是多少。

控件数组共享同样的事件过程,这在希望若干控件共享代码时非常方便。例如,控件数组 Command1 有 3 个命令按钮,不论单击哪个命令按钮,都会调用同 1 个事件过程 Command1_Click。

为了区分控件数组中的各个元素,VB 会把下标值传送给 1 个过程。例如,单击上述控件数组中的任意 1 个命令按钮时,调用的事件过程如下:

```
Private Sub Command1_Click(Index As Integer)
```

```
...
```

```
End Sub
```

在该事件过程中,经常使用分支语句对参数 Index 进行分支处理,以便对不同按钮按下加以区分。

在设计时,采用控件数组添加控件比直接向窗体添加多个相同类型的控件所消耗的资源要少。

5.5.2 控件数组的创建方法

由于控件数组是针对控件创建的,因此与普通数组的定义方法不一样。可以在设计时创建控件数组,也可以在运行时添加控件数组。

1. 在设计时创建控件数组

在设计时,可以通过以下 3 种方法创建控件数组。

(1) 方法 1

将多个控件取相同的名称:

1) 依次绘制作为数组元素的各个控件,并保证它们为同一类型的控件。

2) 单击要包含到数组中的某个控件,将其激活,在属性窗口中,将其 Name 属性设置为控件数组名(也可使用其原有的属性值)。该控件将成为控件数组中的第一个元素。

3) 把第二个控件的 Name 属性改成同一个名称。这时,VB 会显示一个对话框,询问是否要创建控件数组。单击“是”按钮,将创建控件数组;单击“否”按钮,则取消创建操作。若创建成功,第一个控件具有索引值 0,第二个控件具有索引值 1,将后续控件 Name 属性改成同一名称时,不再出现询问对话框,而将其直接加入控件数组中,其索引号依加入次序逐个递增。

(2) 方法 2

复制现有控件:

- 1) 绘制或选择要作为控件数组的第一个控件, 将其激活。
- 2) 执行“编辑”菜单中的“复制”命令(也可单击标准工具栏的“复制”按钮或者按 Ctrl + C 快捷键), 将该控件放入剪贴板中。
- 3) 执行“编辑”菜单中的“粘贴”命令(也可单击标准工具栏的“粘贴”按钮或者按 Ctrl + V 快捷键)。这时, VB 同样会显示一个对话框, 询问是否要创建控件数组。单击“是”按钮, 将创建控件数组; 单击“否”按钮, 则取消创建操作。若创建成功, 绘制的第一个控件具有索引值 0, 而新粘贴的控件具有索引值 1, 再次执行“粘贴”命令时, 不再出现询问对话框, 而将其直接加入控件数组中, 其索引号依加入次序逐个递增。

采用方法 2 向控件数组添加控件时, 大多数可视属性, 例如宽度、高度和颜色, 将从数组中的第一个控件复制到新控件中。

(3) 方法 3

设置控件的 Index 属性值:

- 1) 绘制或选择要作为控件数组的第一个控件, 将其激活。
- 2) 在属性窗口中直接指定一个 Index 属性值(如设置为 0)。
- 3) 采用以上两种方法之一添加数组中的其他控件, 这时不再出现对话框询问是否要创建控件数组。

控件数组建立后, 可以修改 Index 属性值以调整相应控件在控件数组中的位置, 但必须保证同一个控件数组中各元素的 Index 属性值是唯一的。

2. 在运行时添加控件数组

在运行时, 可以通过 Load 方法创建或添加控件数组, 通过 Unload 方法删除数组中的某个控件。

- 1) 绘制或选择要作为控件数组的第一个控件, 将其激活, 在属性窗口中直接指定一个 Index 属性值(如设置为 0), 表示该控件为数组, 这是创建的第 1 个元素。
- 2) 在编程时通过 Load 方法, 添加其余若干个元素, 也可通过 Unload 方法, 删除某个添加的元素。
- 3) 对于每个新添加的控件, 通过 Left 和 Top 属性确定其在窗体的位置, 并将 Visible 属性设置为 True。

【例 5-1】 利用 Load、Unload 方法在程序运行时添加、删除控件数组。在窗体上建立两个命令按钮 Command1、Command2, 将 Command1 的 Caption 属性设置为“添加”, Command2 的 Caption 属性设置为“删除”。再在窗体上建立一个标签控件 Label1, 并在属性窗口中将其 Index 属性设置为 0, AutoSize 属性设置为 True。编写程序代码如下:

```
Dim num As Integer
Private Sub Command1_Click()
    num = num + 1
    If num > 5 Then Exit Sub
    Load Label1(num)
    Label1(num).Left = Label1(num - 1).Left + 500
```

```

Label1(num).Visible = True
End Sub

Private Sub Command2_Click()
    If num > 5 Then num = 5
    If num = 0 Then Exit Sub
    Unload Label1(num)
    num = num - 1
End Sub

Private Sub Label1_Click(Index As Integer)
    Label1(Index).Caption = Index
End Sub

```

“添加”命令按钮的事件过程 Command1_Click 用来增加标签控件，每单击一次命令按钮，用 Load 方法为控件数组 Label1 添加一个元素。新添加的元素位于原控件的右侧，其 Visible 属性被设置为 True。控件数组最大下标值为 5，因此最多可以增加 5 个标签控件，超过 5 个后，将通过 Exit Sub 语句退出该事件过程。

“删除”命令按钮的事件过程 Command2_Click 用来删除标签控件，每单击一次命令按钮，用 Unload 方法删除控件数组 Label1 中的一个元素，被删除的元素是当前索引值最大的元素。最多可以删除 5 个标签控件，超过之后 (num = 0)，将通过 Exit Sub 语句退出该事件过程，因为在设计状态添加的控件数组元素不能用 Unload 方法删除。

标签控件的事件过程 Label1_Click 将被单击的标签控件的标题设置成该控件在数组中的索引值。

5.6 数组的应用

【例 5-2】单击窗体，将 10 名学生的成绩通过 InputBox 输入框依次输入。统计这 10 名学生的成绩分布情况，并将统计结果在窗体上打印出来。成绩统计程序运行结果如图 5-1 所示。

统计成绩分布情况，即统计 0~9、10~19、20~29、30~39、40~49、50~59、60~69、70~79、80~89、90~99 以及 100 分这 11 个成绩段中每一段共有多少名学生。可以建立一个包含 11 个元素的数组，每一个数组元素，对应一个成绩段的学生人数，对输入的学生成绩进行分段计数，完成统计。分段计数时，当然可以采用分支选择语句对输入的成绩进行分类处理，但这样做程序代码稍嫌繁琐。一个简单的办法是，在输入的成绩与作为成绩计数器的数组元素之间建立一种映射关系，可以通过这种映射关系将输入的成绩直接映射到其对应成绩段的计数器数组元素，完成对该计数器元素的计数操作。通过观察可以发现，如果令数组

100-100	3
90-99	2
80-89	2
70-79	2
60-69	1
50-59	0
40-49	0
30-39	0
20-29	0
10-19	0
0-9	0

图 5-1 成绩统计程序运行结果

下标从 0 开始，下标为 0 的数组元素用于统计 0~9 分这一段的成绩，下标为 1 的数组元素用于统计 10~19 分这一段的成绩，依此类推，下标为 10 的数组元素用于统计 100 分这一段的成绩，则输入成绩与对应数组元素的映射关系为：输入成绩被 10 整除后将得到对应数组

元素的下标。通过下标,即可令相应的数组元素加1,完成对该成绩的计数。

程序代码如下:

```
Private Sub Form_Click()
    Dim N(10) As Integer, i As Integer, Score As Integer
    Const m = 10
    For i = 1 To m
        Score = InputBox("请输入第" & Str(i) & "个学生的成绩")
        Score = Int(Score/10) '也可写成 Score = Score \ 10
        N(Score) = N(Score) + 1
    Next i
    Cls
    Print 100; "----"; 100; N(10)
    For i = 9 To 0 Step -1
        Print 10 * i; "----"; 10 * i + 9; ";"; N(i)
    Next i
End Sub
```

【例 5-3】 输入一组不重复的数据,找出最大值、最小值及其位置。求最大值、最小值程序运行结果如图 5-2 所示。

本例是求数组的最大值、最小值问题,该问题的解决办法是:

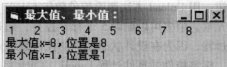


图 5-2 求最大值、最小值程序运行结果

1) 建立两个变量 Max 和 Min, 分别用于存放最大值和最小值, 令两个变量的初值均为数组中的第一个元素的值。如果要找出最大值和最小值的位置, 还要建立两个变量 MaxPos 和 MinPos, 分别用于存放最大值和最小值所对应的数组元素的索引值, 并令其初值均为数组第一个元素的索引值。

2) 依次将 Max 和 Min 与数组中其余元素进行比较, 如果数组中某个元素的值大于变量 Max 的值, 则用该数组元素的值替换 Max 变量原有的值, 并将该数组元素的索引值赋予变量 MaxPos; 如果数组中某个元素的值小于变量 Min 的值, 则用该数组元素的值替换 Min 变量原有的值, 并将该数组元素的索引值赋予变量 MinPos。所有数据比较完成后, Max 中存放的数据即为整个数组的最大值, 其对应的数组元素的位置(索引值)就存放在 MaxPos 中; Min 中存放的数据即为整个数组的最小值, 其对应的数组元素的位置(索引值)就存放在 MinPos 中。

以上算法同样适用于多维数组, 所不同的是要建立更多的变量用于存放最大值和最小值的位置索引。

程序代码如下:

```
Private Sub Form_Click()
    Const length = 8 '定义常量 length 作为数组最大下标
    Dim A(length) As Integer, i As Integer, Max As Integer, _
        MaxPos As Integer, Min As Integer, MinPos As Integer
    For i = 1 To length '通过键盘输入给数组赋值
```

```

A(i) = InputBox("输入第"&i&"个数据")
Print A(i);Space(5)
Next i
Max = A(1);MaxPos = 1      '设数组第一个元素为最大值
Min = A(1);MinPos = 1      '设数组第一个元素为最小值
For i = 2 To length
    If Max < A(i) Then      '找到新的最大值,记录其值和位置
        Max = A(i)
        MaxPos = i
    ElseIf Min > A(i) Then
        Min = A(i)
        MinPos = i
    End If
Next i
Print
Print "最大值 x =" & Max & ",位置是" & MaxPos
Print "最小值 x =" & Min & ",位置是" & MinPos
End Sub

```

【例 5-4】 生成 10 个随机整数存放到一维数组 D 中,并将其按照索引值的顺序和倒序依次在窗体上进行输出。

要顺序输出数组元素,只需按照索引值顺序(由小到大)遍历数组元素进行输出即可。同理,要倒序输出数组元素,只需按照索引值倒序(由大到小)遍历数组元素进行输出即可。

数组的遍历程序运行结果如图 5-3 所示,代码如下:

```

Private Sub Form_Click()
    Const length = 10
    Dim D(length) As Integer,i As Integer
    Randomize Timer      '设置随机化种子,保证每组数据不重复
    Cls                  '清空窗体
    For i = 1 To length  '给数组赋值并输出
        D(i) = 100 * Rnd;Print D(i);
    Next i
    Print:Print          '换行
    For i = length To 1 Step -1 '倒序输出
        Print D(i);
    Next i
End Sub

```

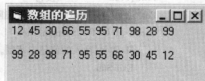


图 5-3 数组的遍历程序运行结果

【例 5-5】 生成 10 个随机整数存放到一维数组 D 中,并将其在窗体上显示出来。单击窗体,在弹出的输入框中输入要删除的数组元素的位置值,删除相应的数组元素,并将删除后的数组内容在窗体上显示出来。

要删除数组 D 中指定的 Pos 位置的元素, 实际上是将指定位置元素值后的所有元素依次向前移动一位, 对于有 length 个元素的数组, 其删除过程如下:

$$D(Pos) = D(Pos + 1)$$

$$D(Pos + 1) = D(Pos + 2)$$

$$\dots$$

$$D(length - 1) = D(length)$$

程序运行时, 输入要删除的数的位置, 完成删除操作。如图 5-4 显示了将数组中第一个元素删除后的结果。

程序代码如下:

```
Private Sub Form_Click()
    Const length = 10
    Dim D(length) As Integer, i As Integer, Pos As Integer
    Randomize Timer '设置随机化种子, 保证每组数据不重复
    Cls '清空窗体
    For i = 1 To length '给数组赋值并输出
        D(i) = 100 * Rnd; Print D(i);
    Next i
    Print; Print '换行
    Pos = Val(InputBox("请输入要删除的元素的位置:"))
    If Pos < 1 Or Pos > length Then
        MsgBox "输入的值超出范围, 请重新输入!"
        Exit Sub
    End If
    For i = Pos To length - 1 '删除数组元素
        D(i) = D(i + 1)
    Next i
    For i = 1 To length - 1 '输出删除后的数组
        Print D(i);
    Next i
End Sub
```

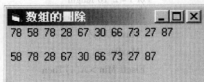


图 5-4 数组的删除程序运行结果

【例 5-6】 生成 10 个随机整数存放到一维数组 D 中, 并将其在窗体的文本框中显示出来, 然后向数组中的指定位置插入一个指定的数, 将插入后的结果显示于窗体的另一个文本框中。如果指定位置小于或等于零, 则将指定的数插在数组的第一个位置; 如果指定的位置大于现有数据的个数, 则将指定的数插在数组的最后一个位置。

将某数 Num 插在数组 D 中指定的位置 Pos, 即将数组 D 中原 Pos 位置的元素到最后一个元素全部向后移动一个位置, 而新的数作为数组的 Pos 位置的元素。所以, 每次插入都将使数组的总元素个数增加 1。

要对数组中原 Pos 位置的元素到最后一个元素全部向后移动一个位置, 需要从后面的元素开始逐个向前作移动操作, 即执行以下操作:

$$D(length + 1) = D(length)$$

```
D(length) = D(length - 1)
```

```
...
```

```
D(Pos + 1) = D(Pos)
```

然后, 将 Num 作为数组的第 Pos 位置的元素, 即执行以下操作:

```
D(Pos) = Num
```

数组的插入程序设计界面如图 5-5a 所示。文本框 Text1 用于显示原数组元素, 文本框 Text2 用于显示插入后的数组元素, 将这两个文本框设计成带水平滚动条的文本框, 以便能水平显示多个数据。插入的数据和插入的位置分别由文本框 Text3、Text4 输入。图 5-5b 显示了在第 2 个位置插入数 69 后的结果。

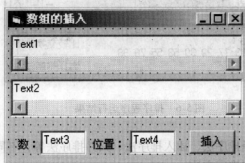
在窗体模块的声明段声明如下:

```
Option Base 1
```

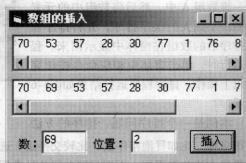
```
Dim length As Integer, D() As Integer
```

在窗体的 Load 事件过程中生成 10 个 1~100 之间的随机整数, 显示在文本框 Text1 中, 代码如下:

```
Private Sub Form_Load()  
    Text1.Text = ""  
    Text2.Text = ""  
    Text3.Text = ""  
    Text4.Text = ""  
    length = 10  
    ReDim D(length)  
    For I = 1 To length  
        D(I) = Int(Rnd * 100)  
        Text1.Text = Text1.Text & Str(D(I)) & " "  
    Next I  
End Sub
```



a)



b)

图 5-5 数组的插入

a) 设计界面 b) 运行界面

单击“插入”按钮将数据插入到数组中, 并显示插入后的结果。“插入”按钮 Command1 的 Click 事件过程如下:

```
Private Sub Command1_Click()
```

```
Text2.Text = ""
```

```
Num = Val(Text3.Text)
```

```
Pos = Val(Text4.Text)
```

```
length = length + 1 '将数组的总个数增加 1
```

```
ReDim Preserve D(length) '定义动态数组 D, 并保留数组中原有的值
```

```
Select Case Pos
```

```
Case Is <= 0 '如果指定的位置值小于或等于零, 将数插在数组的第一个位置
```

```
For I = length To 2 Step -1
```

```
D(I) = D(I - 1)
```

```
Next I
```

```
D(1) = Num
```

```
Case Is > length
```

```
'如果指定位置大于原有数据总个数, 将数插在数组的最后一个位置
```

```
D(length) = Num
```

```
Case Else '如果给定一个正确的插入位置, 将数插在数组的指定位置
```

```
For I = length To Pos + 1 Step -1
```

```
D(I) = D(I - 1)
```

```
Next I
```

```
D(Pos) = Num
```

```
End Select
```

```
'显示插入后的结果
```

```
For I = 1 To length
```

```
Text2.Text = Text2.Text & Str(D(I)) & " "
```

```
Next I
```

```
End Sub
```

【例 5-7】 生成 10 个随机整数存放于一维数组 A 中, 然后将数组中的元素按照由小到大的次序排序, 并将排序前和排序后的数组在窗体中显示出来。程序运行时, 单击窗体中相应命令按钮完成以上操作。排序程序运行结果如图 5-6 所示。

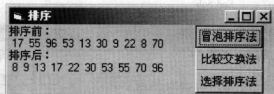


图 5-6 排序程序运行结果

本例是一个排序问题, 排序的方法

有很多种, 比如冒泡排序法、比较交换法、选择排序法、插入排序法、归并排序法等, 不同的排序方法效率不同。下面分别介绍几种常用的排序方法。

(1) 冒泡排序法

冒泡排序法是最简单的排序方法。这种方法的基本思想是, 将待排序的元素看做是竖着排列的“气泡”, 较小的元素比较轻, 从而要往上浮, 较大的元素比较重, 从而要往下沉。在冒泡排序算法中, 要对这个“气泡”序列扫描若干遍。所谓扫描一遍, 就是自底向上检查一遍这个序列, 并时刻注意两个相邻的元素的顺序是否正确。如果发现两个相邻元素的顺

序不对,即“轻”的元素在下面,就交换它们的位置。显然,扫描一遍之后,“最轻”的元素就浮到了最高位置;扫描两遍之后,“次轻”的元素就浮到了次高位置。在作第二遍扫描时,由于最高位置上的元素已是“最轻”元素,所以不必检查。一般地,第 i 遍处理时,不必检查第 i 高位置以上的元素,因为经过前面 $i-1$ 遍的处理,它们已正确地排好序。

例如,数组 A 包含6个元素,分别是11、16、30、9、8、19,用冒泡排序法对数组进行排序,过程如下:

1) 第1遍扫描,如图5-7a所示,经过5次两两比较,最小的元素值8,像气泡一样升到了最上面,找到了自己在排序中的位置。下一遍扫描时,它便不需再参与(在图中用灰色表示,下同)。

2) 第2遍扫描,如图5-7b所示,经过4次两两比较,次小的元素值9,像气泡一样升到了自己在排序中的位置。下一遍扫描时,它便不需再参与。

3) 第3遍扫描,如图5-7c所示,经过3次两两比较,元素值11找到了自己在排序中的位置。下一遍扫描时,它便不需再参与。

4) 第4遍扫描,如图5-7d所示,经过2次两两比较,元素值16找到了自己在排序中的位置。下一遍扫描时,它便不需再参与。

5) 第5遍扫描,如图5-7e所示,经过1次两两比较,元素值19找到了自己在排序中的位置。至此,扫描结束,剩余的元素30是最大元素。

总结以上排序过程可以得出结论,对于 N 个数进行冒泡排序,需要扫描 $N-1$ 遍,每一遍需要两两比较的次数为 N 减去扫描遍数。如6个数排序,需要扫描5遍,第3遍需要两两比较的次数为 $(6-3)$ 次=3次。

冒泡排序的程序实现如下:

```
Private Sub Command1_Click()  
    Const N = 10  
    Dim A(N) As Integer, i As Integer, j As Integer, _  
        t As Integer  
    Randomize Timer '设置随机化种子,保证每组数据不重复  
    Cls '清空窗体  
    Print "排序前:"  
    For i = 1 To N '给数组赋值并输出  
        A(i) = Int(100 * Rnd): Print A(i);  
    Next i  
    Print '换行  
    '冒泡排序法  
    For i = 1 To N - 1  
        For j = 1 To N - i  
            If A(N + 1 - j) < A(N - j) Then  
                t = A(N + 1 - j)  
                A(N + 1 - j) = A(N - j)  
                A(N - j) = t  
            End If  
        Next j  
    Next i
```

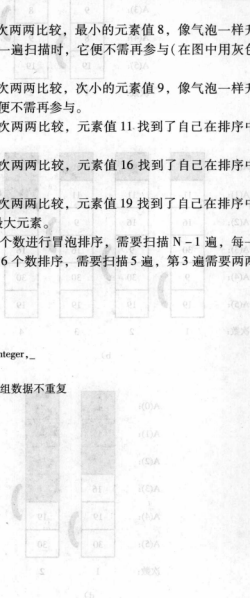


图 5-7 冒泡排序过程

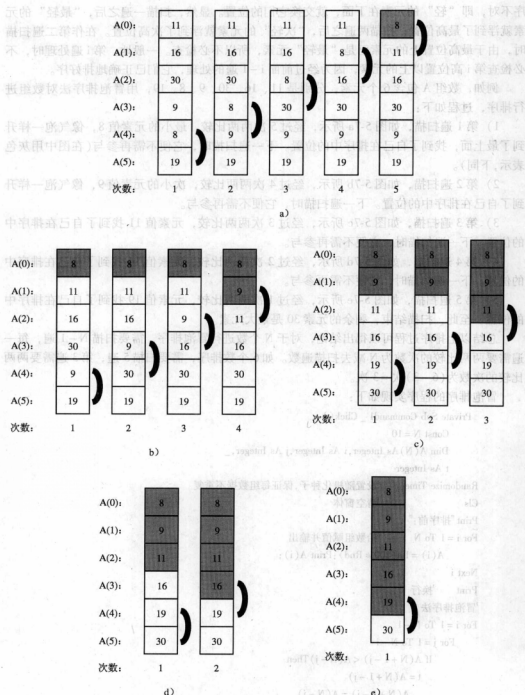


图 5-7 冒泡排序过程

a) 第 1 遍扫描 b) 第 2 遍扫描 c) 第 3 遍扫描 d) 第 4 遍扫描 e) 第 5 遍扫描

```

Next i
Print "排序后:"
For i = 1 To N    '输出排序后的数组
    Print A(i);
Next i
End Sub

```

(2) 比较交换法

对含有 N 个元素的数组 A 的比较交换法步骤如下:

1) 第 1 遍扫描, 将数组中第 1 个元素和其余 $N-1$ 个元素依次比较, 如果在其余 $N-1$ 个元素中发现有小于第 1 个元素的, 则用它与第 1 个元素交换。这一遍共比较了 $N-1$ 次, 找到了第 1 个最小的元素, 将它放置到了 $A(1)$ 中, 不再参加后面的扫描。

2) 第 2 遍扫描, 将数组中第 2 个元素和其余 $N-2$ 个元素依次比较, 如果在其余 $N-2$ 个元素中发现有小于第 2 个元素的, 则用它与第 2 个元素交换。这一遍共比较了 $N-2$ 次, 找到了第 2 个最小的元素, 将它放置到了 $A(2)$ 中, 不再参加后面的扫描。

3) 第 i 遍扫描, 将数组中第 i 个元素和其余 $N-i$ 个元素依次比较, 如果在其余 $N-i$ 个元素中发现有小于第 i 个元素的, 则用它与第 i 个元素交换。这一遍共比较了 $N-i$ 次, 找到了第 i 个最小的元素, 将它放置到了 $A(i)$ 中, 不再参加后面的扫描。

共重复 $N-1$ 遍扫描。

比较交换法的程序实现如下:

```

Private Sub Command2_Click()
    Const N = 10
    Dim A(N) As Integer, i As Integer, j As Integer, _
    t As Integer
    Randomize Timer    '设置随机化种子, 保证每组数据不重复
    Cls                '清空窗体
    Print "排序前:"
    For i = 1 To N    '给数组赋值并输出
        A(i) = Int(100 * Rnd); Print A(i);
    Next i
    Print "换行"
    '比较交换法排序
    For i = 1 To N - 1
        For j = i + 1 To N
            If A(j) < A(i) Then '如果其余元素中有比当前元素小的, 则交换
                t = A(i)
                A(i) = A(j)
                A(j) = t
            End If
        Next j
    Next i
End Sub

```



```

Next i
Print "排序后:"
For i = 1 To N    '输出排序后的数组
    Print A(i);
Next i
End Sub

```

(3) 选择排序法

对含有 N 个元素的数组 A 的选择排序法步骤如下:

1) 第 1 遍扫描, 将数组中第 1 个元素和其余 $N-1$ 个元素依次比较, 找出第 1 个元素到第 N 个元素中的最小值, 记下该元素的下标值 p 。这一遍扫描结束后, 将 $A(1)$ 与 $A(p)$ 交换。此遍共比较了 $N-1$ 次, 找到了第 1 个最小的元素, 将它放置到了 $A(1)$ 中, 不再参加后面的扫描。

2) 第 2 遍扫描, 将数组中第 2 个元素和其余 $N-2$ 个元素依次比较, 找出第 2 个元素到第 N 个元素中的最小值, 记下该元素的下标值 p 。这一遍扫描结束后, 将 $A(2)$ 与 $A(p)$ 交换。此遍共比较了 $N-2$ 次, 找到了第 2 个最小的元素, 将它放置到了 $A(2)$ 中, 不再参加后面的扫描。

3) 第 i 遍扫描, 将数组中第 i 个元素和其余 $N-i$ 个元素依次比较, 找出第 i 个元素到第 N 个元素中的最小值, 记下该元素的下标值 p 。这一遍扫描结束后, 将 $A(i)$ 与 $A(p)$ 交换。此遍共比较了 $N-i$ 次, 找到了第 i 个最小的元素, 将它放置到了 $A(i)$ 中, 不再参加后面的扫描。

...

共重复 $N-1$ 遍扫描。

选择排序法的程序实现如下:

```

Private Sub Command3_Click()
    Const N = 10
    Dim A(N) As Integer, i As Integer, j As Integer, _
        t As Integer, p As Integer
    Randomize Timer    '设置随机化种子, 保证每组数据不重复
    Cls                '清空窗体
    Print "排序前:"
    For i = 1 To N    '给数组赋值并输出
        A(i) = Int(100 * Rnd): Print A(i);
    Next i
    Print "换行"
    '选择排序法排序
    For i = 1 To N - 1
        p = i
        For j = i + 1 To N
            If A(j) < A(p) Then '如果其余元素中有更小的, 则对其进行标记
                p = j
            End If

```

```

Next j
If p < > i Then
    t = A(i)
    A(i) = A(p)
    A(p) = t
End If
Next i
Print "排序后:"
For i = 1 To N    '输出排序后的数组
    Print A(i);
Next i
End Sub

```

【例 5-8】 生成 10 个随机整数存放到一维数组 D 中，然后将该数组倒置，并将倒置前和倒置后的数组在窗体中显示出来。

一维数组的倒置是指将数组中的第 1 个元素和最后 1 个元素进行交换，第 2 个元素和倒数第 2 个元素进行交换，第 3 个元素和倒数第 3 个元素进行交换……以此类推，将下标和等于下界和上界之和的各对元素全部两两交换。

含有奇数个元素的一维数组的倒置如图 5-8a 所示，含有偶数个元素的一维数组的倒置如图 5-8b 所示。

倒置前:	89	90	82	36	78	55	66
倒置后:	66	55	78	36	82	90	89

a)

倒置前:	2	1	6	3	8	5
倒置后:	5	8	3	6	1	2

b)

图 5-8 数组倒置

a) 含奇数个元素的维数组倒置 b) 含偶数个元素的维数组倒置

一维数组倒置的算法如下：

```
For i = 下界 To (上界 - 下界 + 1) \ 2
```

```
    t = D(i)
```

```
    D(i) = D(上界 + 下界 - i)
```

```
    D(上界 + 下界 - i) = t
```

```
Next i
```

程序界面只需要一个窗体，单击窗体在 Form_Click 事件过程中完成 10 个随机整数的生成、显示、数组倒置运算、显示结果等操作。数组的倒置程序运行结果如图 5-9 所示。

程序代码如下：

```
Private Sub Form_Click()
```

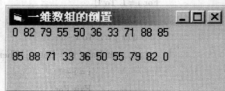


图 5-9 数组的倒置程序运行结果


```

Const length = 10
Dim D(length) As Integer, i As Integer, t As Integer
Randomize Timer '设置随机化种子,保证每组数据不重复
Cls '清空窗体
For i = 1 To length '给数组赋值并输出
    D(i) = 100 * Rnd; Print D(i);
Next i
Print; Print '换行
For i = 1 To length \ 2 '此循环完成数组倒置
    t = D(i)
    D(i) = D(length + 1 - i)
    D(length + 1 - i) = t
Next i
For i = 1 To length '输出倒置后的数组
    Print D(i);
Next i
End Sub

```

【例 5-9】产生 50 个不重复的 3 位随机整数,并按每行 10 列的格式输出。

本例涉及的是数组元素的换行输出,对于要求一行输出 M 个数组元素然后换行继续输出的问题,假设数组名为 Data,其解决办法如下:

```

For i = 下界 To 上界
    Print Data(i);
    If (i - 下界 + 1) Mod M = 0 Then Print
Next i

```

为了更加便于理解,本例程序中特意将数组下界定为 -20,上界定为 29,共 50 个元素。程序中利用一个嵌套循环来保证产生不重复的 3 位随机整数。数组的换行输出程序的运行结果如图 5-10 所示。

程序代码如下:

```

Private Sub Form_Click()
    Const L = -20, U = 29, M = 10
    Dim Data(L To U) As Integer, i As Integer, j As Integer
    Randomize Timer '设置随机化种子,保证每组数据不重复
    For i = L To U '循环产生 100 个数据
        Data(i) = Int(Rnd() * 900) + 100
        For j = L To i - 1 '与已经产生的数据比较
            If Data(i) = Data(j) Then '数据已存在则舍弃,重新产生
                i = i - 1
            Exit For '提前退出数据比较的循环
        Next j
    Next i

```

189	896	255	837	130	880	208	583	504	243
892	640	527	718	760	985	773	866	584	932
205	477	152	945	636	695	590	179	610	769
607	542	310	430	162	183	391	696	763	710
138	392	588	227	349	107	624	994	141	970

图 5-10 数组的换行输出程序运行结果

```

Next i
Cls
For i = L To U
    Print Data(i);
    If (i - L + 1) Mod M = 0 Then Print
Next i
End Sub

```

【例 5-10】统计输入的任意个数的平均值。

本例利用动态数组 A(), 实现对任意数据的输入统计操作。程序首先弹出输入框, 让用户键入要输入的数据个数 N, 然后依次完成数据的输入, 每输入一个数据, 将数据加入到累加和 Sum 中。数据输入完毕, 将求得的累加和 Sum 除以数据个数 N, 得到统计平均值。动态数组程序运行结果如图 5-11 所示。

程序代码如下:

```

Private Sub Form_Click()
    Dim N As Integer, A() As Single, i As Integer, Sum As Single
    N = InputBox("输入几个数? ")
    ReDim A(1 To N) '动态数组的重定义
    Cls '清空窗体
    For i = 1 To N
        A(i) = InputBox("输入第" + Str(i) + "个数")
        Sum = Sum + A(i) '计算累加和
        Print A(i); Space(5);
    Next i
    Sum = Sum / N '求平均值
    Print '换行
    Print N; "个数的平均值为"; Sum
End Sub

```

【例 5-11】输出斐波那契级数的前 20 项。

斐波那契级数是意大利中世纪数学家斐波那契提出的一个数列, 其定义如下:

$$\text{Fib}(n) = \begin{cases} 1 & (n=1) \\ 1 & (n=2) \\ \text{Fib}(n-1) + \text{Fib}(n-2) & (n>2) \end{cases}$$

由公式可知: 斐波那契级数中, 前两项均为 1, 从第三项开始, 每一项均为前面相邻两项之和。因此, 要产生前 20 项斐波那契级数, 只需为相应数组的前 2 项赋值为 1, 其余各项对应的数组元素按照公式依次求出即可。斐波那契级数程序运行结果如图 5-12 所示。

程序代码如下:

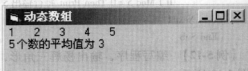


图 5-11 动态数组程序运行结果

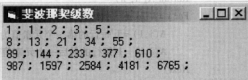


图 5-12 斐波那契级数程序运行结果

```
Private Sub Form_Click()
```

```
    Dim F(20) As Integer, I As Integer
```

```
    Cls '清空窗体
```

```
    F(1) = 1; F(2) = 1 '第一、第二项为 1
```

```
    For I = 3 To 20 '第三项起每项为前两项之和
```

```
        F(I) = F(I-2) + F(I-1)
```

```
    Next I
```

```
    For I = 1 To 20 '在窗体上输出
```

```
        Print F(I); " ";
```

```
        If I Mod 5 = 0 Then Print "每行输出 5 个数组元素"
```

```
    Next I
```

```
End Sub
```

【例 5-12】编写程序，输出杨辉三角形，如图 5-13 所示。

杨辉三角形，又称贾宪三角形、帕斯卡三角形，是中国北宋数学家贾宪（约 1050 年）首先发现的。南宋数学家杨辉在《详解九章算法》（1261 年）一书中对此曾有记载。法国数学家 B. 帕斯卡在 1654 年也发现了这个三角形。杨辉三角形是二项式系数在三角形中的一种几何排列。由图 5-13 的排列格式可以看出，杨辉三角形每一行的第一列和最后一列均为 1，其余各项的值都是其上一行中前一列元素与本列元素之和。由此可得算法如下：

$$A(i, j) = A(i-1, j-1) + A(i-1, j)$$

建立一个动态二维数组，用来存放生成的各项数值。令二维数组的行、列下标下界均为 1，则二维数组中第 i 行实际只是用到前 i 列数组元素，这可以通过一个双重循环来实现。外层循环控制行的访问，内层循环控制列的访问。

程序代码如下：

```
Private Sub Form_Click()
```

```
    Dim A()
```

```
    Dim i As Integer, n As Integer
```

```
    n = Val(InputBox("请输入欲输出的杨辉三角形的行数:"))
```

```
    Cls '清空窗体
```

```
    ReDim A(n, n)
```

```
    For i = 1 To n
```

```
        A(i, 1) = 1 '每行的第一个元素值为 1
```

```
        A(i, i) = 1 '每行的最后一个元素值为 1
```

```
    Next
```

```
    For i = 1 To n '行控制
```

```
        Print Tab(5);
```



图 5-13 杨辉三角形

```

For j=1 To i '列控制
    If j=1 Or j=i Then
        A(i,j)=1
    Else
        A(i,j)=A(i-1,j-1)+A(i-1,j)
    End If
    Print Format(A(i,j),"! @@@@");
Next
Print
Next
End Sub

```

以上程序在 Format 语句的格式描述中使用了字符占位符“@”。该符号表示要在相应的位置显示字符或是空格。如果字符串在格式字符串中@的位置有字符存在,那么就显示出来;否则,就在那个位置上显示空格。字符占位符规定由右向左填充字符。这里在格式描述的左侧使用“!”表示强制由左向右填充字符。

【例 5-13】 建立一个 5 行 5 列的矩阵,其两条对角线上的元素值均为 1,其余元素均为 0,将其在窗体上输出,如图 5-14 所示。

矩阵中的每个数据在矩阵中所处的位置由行号和列号决定,可以使用一个二维数组直观地表示矩阵中的每一个元素。例如,用二维数组 s 表示矩阵,第一个下标表示矩阵中数据所在的行号,第二个下标表示列号,因此矩阵中的第 i 行第 j 列元素表示为 s(i,j)。而对于一个 N 行 N 列的矩阵,其主对角线元素的行下标与列下标相同,次对角线元素的行下标与列下标之和为 N+1。

程序代码如下:

```

Private Sub Form_Click()
    Dim s(1 To 5,1 To 5) As Integer, i As Integer, j As Integer
    Cls      '清空窗体
    For i=1 To 5
        For j=1 To 5
            If i=j Or i+j=6 Then '如果为主、次对角线元素
                s(i,j)=1
            Else
                s(i,j)=0 '如果为其余元素
            End If
            Print s(i,j); '输出
        Next j
    Next i
    Print
End Sub

```

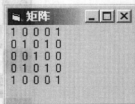


图 5-14 主对角线和次对角线为 1 的方阵

【例 5-14】 建立一个类似国际象棋的棋盘,要求黑白交替,运行时单击某个棋格,会



改变颜色并显示其行列号。

本例可用控件数组实现，黑白交替的棋格可以用一组标签控件实现，当用户单击某个棋格时，引起控件数组的 Click 事件过程被执行，在事件过程中可以通过 Index 参数定位被单击的棋格对应的控件数组的元素，将该元素的背景色设置成其他颜色，然后求出该控件的行列号，并加以显示。

在窗体上建立一个标签控件 Label1，并在属性窗口中将其 Index 属性设置为 0，Width 和 Height 属性均设置为 500。同时，将窗体的 ScaleWidth 和 ScaleHeight 属性均设置为 5000。窗体的绘图区共可容纳 100 个标签控件，分为 10 行 10 列。

棋盘程序运行结果如图 5-15 所示。



图 5-15 棋盘程序运行结果

Form_Load 事件过程中利用 Load 方法完成控件数组元素的添加，并设立一个逻辑变量 IsWhite 用来控制各控件元素的背景色设置。IsWhite 为 True 时，将背景色设置为白色；IsWhite 为 False 时，将背景色设置为黑色。

程序代码如下：

```
Private Sub Form_Load()  
    Dim W As Integer, H As Integer  
    Dim IsWhite As Boolean  
    W = Label1(0).Width  
    H = Label1(0).Height  
    Label1(0).Left = 0  
    Label1(0).Top = 0  
    Label1(0).Caption = ""  
    Label1(0).BackColor = vbWhite  
    For i = 1 To 99
```

```
        Load Label1(i)
```

```

Label1(i). Width = W
Label1(i). Height = H
Label1(i). Caption = ""
If i Mod 10 = 0 Then
    IsWhite = Not IsWhite
End If
If IsWhite Then
    Label1(i). BackColor = vbWhite
Else
    Label1(i). BackColor = vbBlack
End If
IsWhite = Not IsWhite
Label1(i). Left = (i Mod 10) * W
Label1(i). Top = (i \ 10) * H
Label1(i). Visible = True
Next i
End Sub
Private Sub Label1_Click(Index As Integer)
    Label1(Index). BackColor = vbRed
    x = Index \ 10 + 1
    y = Index Mod 10 + 1
    MsgBox "您点击的是第" & x & "行,第" & y & "列"
End Sub

```

本例使用控件数组，利用 Label1_Click 事件过程对控件数组全部 100 个元素统一处理，大大简化了程序编码。如果不使用控件数组，则在每一个标签的单击事件过程中都要编写类似的代码。

习 题

5-1 单击窗体，随机生成 10 个 -100 ~ 100 之间的整数存放于一维数组 A 中，统计正数的个数、正数的平均值、负数的个数、负数的平均值。用 Print 方法将数组内容和统计结果在窗体上打印出来。

5-2 统计输入的任意个数的和，将输入的数与它们的和一起在窗体上显示出来。

5-3 输入 N 名学生的成绩，显示于文本框 Text1 中，按成绩从低到高的次序排序，并将结果显示于另一个文本框 Text2 中，N 可以通过输入框任意指定。要求设计界面如图 5-16 所示。

5-4 产生 81 个不重复的 2 位随机整数，并按每行 9 列的格式输出。

5-5 求矩阵每行元素的和及每列元素的和，矩阵内容由用户输入。要求界面如图 5-17 所示。

5-6 设计如图 5-18 所示的界面，创建一选项按

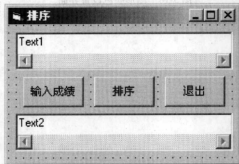


图 5-16 学生成绩排序界面

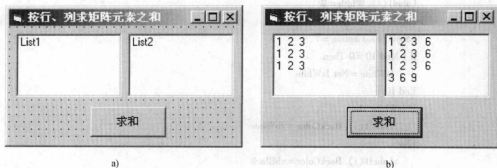


图 5-17 求矩阵每行、每列的和

a) 设计界面 b) 运行界面

钮控件数组 (Option1)。运行时, 当按下某一选项按钮时, 对图形设置相应的填充颜色。图中右侧为 Shape 控件, 在属性窗口中将它 FillStyle 属性设置为 0, 然后可在程序中通过对其 FillColor 属性赋予不同的值, 来改变其颜色。其中, 白色用 vbWhite 赋值, 蓝色用 vbBlue 赋值, 绿色用 vbGreen 赋值, 红色用 vbRed 赋值, 黄色用 vbYellow 赋值。

5-7 设计一个模拟电话机拨号程序, 界面要求如图 5-19 所示。运行时, 当按下某一数字按钮时, 按钮上的数字出现在显示电话号码的文本框内, 按下“清空”按钮则清空文本框中的电话号码。要求使用控件数组实现电话号码的输入。

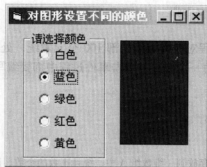
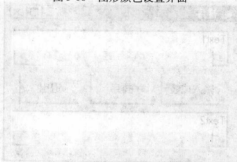


图 5-18 图形颜色设置界面



图 5-19 模拟电话拨号界面



第 6 章 过 程

在程序设计过程中,为各个相对独立的功能模块所编写的一段程序称之为过程。定义过程的主要目的如下:

- 1) 在处理实际问题时,可以使同一个程序段重复使用,减少程序员代码编写的工作量。
- 2) 在程序中调用子过程,可以大大改善程序的结构,把复杂的问题分解成若干个简单问题进行设计。

3) 便于程序的调试,调试单独的单元与调试不包含过程的整个程序相比要容易。

VB 中的过程可分为两类: Sub 过程和 Function 过程。本章将重点介绍这两类过程。

6.1 Sub 过程

在 VB 中,将没有任何返回值的过程定义为 Sub 过程。要在程序中使用 Sub 过程,首先要定义 Sub 过程。

6.1.1 Sub 过程的定义

定义 Sub 过程的格式如下:

[Public/Private][Static]Sub 子过程名([形式参数列表])

<局部变量或常量的定义>

<语句块>

[Exit Sub]

<语句块>

End Sub

说明:

1) Sub 过程以 Sub 开始,以 End Sub 结束,在 Sub 和 End Sub 之间是过程体,用来描述 Sub 过程的功能。

2) Public 定义的过程为公有过程,可被任何过程调用;Private、Static 定义的过程为局部过程,只能在定义此过程的模块中被调用。

3) 子过程名:必须是 VB 中合法的标识符,符合标识符的命名规则。

4) 形式参数列表:用来表示形式参数的类型、个数、位置,列表中可以有多个参数,参数之间用“,”分割。过程中可以没有任何形式参数,但括号不能省略。参数的定义格式如下:

[ByVal|ByRef]变量名[()][As 类型][,...]

ByVal 表示当此过程被调用时,参数是按值传递的;默认或 ByRef 表示参数是按地址传递的。

5) [Exit Sub]是退出 Sub 过程的语句,它通常与条件语句联用,即当满足一定条件时退出 Sub 过程。



【例 6-1】 定义一个交换两个整形变量值的 Sub 过程。

```
Private Sub Swap(a As Integer,b As Integer)
    Dim Temp As Integer
    Temp = a
    a = b
    b = Temp
End Sub
```

对于过程 Swap 的定义也可以写成:

```
Private Sub Swap(ByRef a As Integer,ByRef b As Integer)
```

6.1.2 Sub 过程的创建

Sub 过程通常在窗体模块(.FRM)和标准模块(.BAS)中创建。创建 Sub 过程有两种方式:一是通过定义直接创建;二是利用 VB 中的工具创建。

(1) 方法 1:通过定义直接创建

在窗体的“代码窗口”或标准模块的“代码窗口”中直接按定义格式输入代码来创建 Sub 过程。在窗体模块和标准模块中创建 Sub 过程 Swap,如图 6-1 和图 6-2 所示。

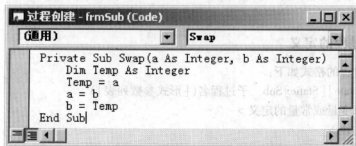


图 6-1 在窗体模块创建 Sub 过程

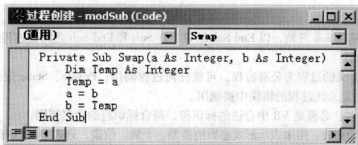


图 6-2 在标准模块中创建 Sub 过程

(2) 方法 2:利用 VB 中的工具创建

1) 在 VB 开发环境下,选择创建 Sub 过程的窗体,并进入“代码窗口”。

2) 依次选择 VB 系统菜单中“工具”→“添加过程(P)...”子菜单项,打开“添加过程”窗口,如图 6-3 所示。

3) 在“添加过程”窗口中,输入所建立过程的名称(Swap);选择过程类型(子过程);定义过程的作用范围(私有的)。

4) 在“添加过程”窗口中,单击“确定”按钮,便建立一个 Sub 过程的结构框架。

```
Private Sub Swap()
```

```
.....
```

```
End Sub
```

5) 若 Sub 过程带有参数,必须在括号中填写参数的定义。过程中的代码由程序员根据创建 Sub 过程的功能要求来编写。

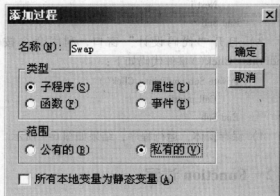


图 6-3 “添加过程”窗口

6.1.3 Sub 过程的调用

Sub 过程的调用格式有两种方式:直接用过程名调用和利用 Call 语句调用。格式分别如下:

- 1) 直接用过程名调用格式:过程名[实际参数列表]。
- 2) 利用 Call 语句调用格式:Call 过程名([实际参数列表])。

说明:

- 1) 直接用过程名调用时必须省略参数两边的()。
- 2) 使用 Call 语句调用时,参数必须放在()内,若没有参数则()可以省略。
- 3) 在调用时实际参数和形式参数的数据类型、个数和顺序必须保持一致;实际参数的类型可以是变量、常量、表达式和数组,当实际参数按引用传递时,形式参数的数据类型只能是变量和数组。

【例 6-2】设计一个过程,用来输出 1~100 中所有偶数。

操作步骤如下:

- 1) 窗体和控件属性参照图 6-4 所示。
- 2) 打开“代码设计”窗口,定义一个 Sub 过程。Sub 过程程序代码如下:

```
Sub printEven()
```

```
Dim i As Integer
```

```
Dim j As Integer
```

```
For i = 1 To 100
```

```
    If i Mod 2 = 0 Then
```

```
        Print i;
```

```
        j = j + 1
```

```
    If j Mod 10 = 0 Then
```

```
        Print
```

```
    End If
```

```
End For
```



图 6-4 输出 1~100 所有偶数



```
Next i
```

```
End Sub
```

3) 打开“代码设计”窗口,利用命令按钮控件的事件代码,调用 Sub 过程。
cmdPrint_Click() 事件代码如下:

```
Private Sub cmdPrint_Click()
```

```
Call printEven
```

```
End Sub
```

4) 保存窗体,运行程序,结果如图 6-4 所示。

6.2 Function 过程

Function 过程也叫函数过程,是过程的另一种形式。在 VB 系统中,函数分为内部函数和外部函数。

内部函数是系统预先编译好的、能完成特定功能的一段程序,如 Abs()、Sqr()、Cos() 等函数;外部函数是用户根据需要用 Function 关键字定义的函数,与内部函数的使用方法一样。

Function 过程和 Sub 过程不同之处是 Function 过程将返回一个值,而 Sub 过程没有返回值。

6.2.1 Function 过程的定义

定义 Function 过程的格式如下:

```
[Public|Private][Static]Function 函数名([形式参数列表])[As 数据类型]
```

```
<局部变量或常量的定义>
```

```
<语句块>
```

```
[函数名 = 返回值]
```

```
[Exit Function]
```

```
<语句块>
```

```
[函数名 = 返回值]
```

```
End Function
```

说明:

1) Function 过程以 Function 开始,以 End Function 结束,在 Function 和 End Function 之间是函数体,用来描述 Function 过程的功能。

2) Public 定义的过程为公有函数过程,可被任何过程调用;Private、Static 定义的 Function 为局部函数过程,只能在定义此函数过程的模块中被调用。

3) 函数过程名:必须是 VB 中合法的标识符,符合标识符的命名规则。

4) 形式参数列表:参数的定义及含义和 Sub 过程相同。

5) [Exit Function] 是退出 Function 过程的语句,它通常与条件语句联用,即当满足一定条件时退出 Function 过程。

6) [函数名 = 返回值] 是用来设定 Function 过程的返回值。无论 Exit Function 还是正常运行结束,Function 过程在退出之前,都要有此语句来设定函数的返回值。

6.2.2 Function 过程的创建

创建 Function 过程同 Sub 过程，Function 过程可以在标准模块和窗体模块中创建。

(1) 方法 1：通过定义直接创建

在窗体的“代码窗口”或标准模块的“代码窗口”中直接按 Function 过程的定义格式输入代码来创建 Function 过程。在窗体模块中创建 Function 过程 EvenSum，如图 6-5 和图 6-6 所示。

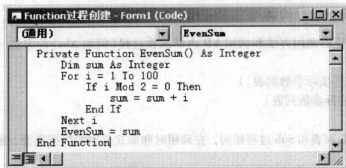


图 6-5 在窗体模块创建 Function 过程

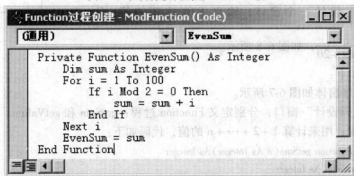


图 6-6 在模块中创建 Function 过程

(2) 方法 2：利用 VB 中的工具创建

1) 在 VB 开发环境下，选择创建 Sub 过程的窗体，并进入“代码窗口”。

2) 依次选择 VB 系统菜单中“工具”→“添加过程(P)...”子菜单项，打开“添加过程”窗口，如图 6-7 所示。

3) 在“添加过程”窗口中，输入所建立过程的名称(EvenSum)；选择过程类型(函数)；定义过程的作用范围(私有的)。

4) 在“添加过程”窗口中，单击“确定”按钮，便建立一个 Function 过程

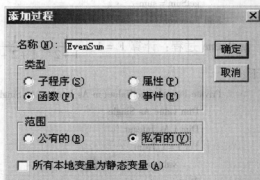


图 6-7 “添加过程”窗口

的定义的结构框架。

```
Private Function EvenSum()
```

```
End Function
```

5) 若 Function 过程带有参数, 必须在括号中填写参数的定义。过程中的代码由程序员根据创建 Function 过程的功能要求来编写。

6.2.3 Function 过程的调用

Function 过程的调用方法和调用 VB 内部函数过程(例如 Abs)的方法一样。语句格式如下:

Call 函数名([实际参数列表])

或函数名[实际参数列表]

说明:

1) 实际参数列表和 Sub 过程相同, 在调用时和形式参数的数据类型、顺序及个数必须匹配。

2) 函数调用必须出现在表达式中, 表示函数过程的返回值。

【例 6-3】设计一个窗体, 输出 F 表达式的值: $F = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \dots + \frac{1}{1+2+3+\dots+19+20}$, 如图 6-8 所示。

操作步骤如下:

1) 创建程序窗体如图 6-7 所示。

2) 在“代码设计”窗口, 分别定义 Function 过程: getSum 和 getValue。

getSum 过程: 用来计算 $1+2+\dots+n$ 的值, 代码如下:

```
Private Function getSum(n As Integer) As Integer
```

```
Dim sum As Integer
```

```
For i = 1 To n
```

```
sum = sum + i
```

```
Next i
```

```
getSum = sum
```

```
End Function
```

getValue 过程: 计算 $F = 1 + \frac{1}{1+2} + \frac{1}{1+2+3} + \dots + \frac{1}{1+2+3+\dots+19+20}$ 的值, 代码

如下:

```
Private Function getValue(m As Integer) As Single
```

```
Dim value As Single
```

```
Dim i As Integer
```

```
For i = 1 To m
```

```
value = value + 1/getSum(i)
```

```
Next i
```

```
getValue = value
```

End Function

3) 定义命令按钮 cmdCompute 的 Click 事件, 并调用 GetValue 过程获得 F 的值。代码如下:

```
Private Sub cmdCompute_Click()
```

```
Label1.Caption = "F 的值:" & GetValue(200)
```

```
End Sub
```

4) 保存窗体, 运行程序, 结果如图 6-8 所示。

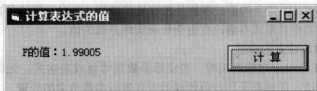


图 6-8 输出 F 的值程序运行结果

6.3 过程的参数传递

参数传递是指在调用带有参数的过程时, 传递给被调用过程的参数, 在 VB 系统中, 参数分为两种: 形式参数和实际参数。

6.3.1 形式参数和实际参数

1. 形式参数

是指在定义通用过程时, 出现在 Sub 或 Function 语句中的过程名后面圆括号内的数, 是用来接收传送给子过程的数据。当形式参数有多个时, 形式参数表中的各个变量之间用逗号分隔。

例如:

```
Private Sub SubDemo(para1 as Integer, para2 as String)
```

```
End Sub
```

此语句定义 Sub 过程 SubDemo, 带有两个形式参数: 整型类型参数 para1, 字符串类型参数 para2。两个参数之间用逗号隔开。

2. 实际参数

实际参数是指在调用 Sub 或 Function 过程时, 写入子过程名或函数名后括号内的参数, 其作用是它们的数据(数值或地址)传送给 Sub 或 Function 过程与其对应的形式参数变量。

实际参数可由常量、变量、表达式、数组名(后加左、右括号, 如 A()) 组成, 实际参数表中各参数间用逗号分隔。

例如:

```
Dim a as Integer
```

```
Dim b as String
```

```
...
```

```
Call SubDemo(a,b)
```

这里的 a 和 b 作为实际参数传递给过程 SubDemo。

6.3.2 按值传递和按地址传递参数

参数的传递方式分为两种：按值传递和按地址传递。

1. 按值传递

使用 ByVal 关键字的形式参数是按值传递的，传递的只是实际参数的副本，形式参数值的改变不会影响实际参数的值。

2. 按地址传递

默认或使用 ByRef 关键字的形式参数是按地址传递的。此时，形式参数和实际参数共用同一内存单元，过程中对形式参数改变也会影响实际参数的值。

3. 参数传递时应注意的问题

1) 在 Sub 和 Function 过程调用时，若实际参数是常量或表达式，无论定义时使用按值传递还是按地址传递，此时都采用按值传递的方式进行参数传递的处理。

2) 当数组作为实际参数传递给过程时，一般采用按地址方式进行传递。

3) 若形式参数定义是按地址传递，调用时想按值传递，可以将实际参数变量加上括号，将其转换成表达式。

【例 6-4】下面程序是在单击窗体上按钮来显示按值和按地址传递时进行加 1 操作的结果。

操作步骤如下：

1) 建立应用程序用户界面和添加控件，并设置各个对象的属性，如图 6-9 所示。

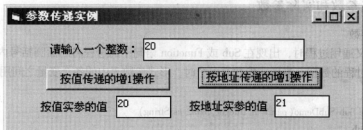


图 6-9 按值传递和按地址传递参数

2) 编写事件代码。

编写 AddByVal 和 AddByRef 通用过程代码：

```
Sub AddByVal(ByVal x As Integer)      '按值传递参数
    x = x + 1
End Sub

Sub AddByRef(ByRef x As Integer)      '按地址传递参数
    x = x + 1
End Sub
```

编写按钮“按地址传递的增 1 操作”的 Click 过程代码如下：

```
Private Sub Command1_Click()
    Dim i As Integer
```

```

i = Val(Text1.Text)
AddByVal i
Text2.Text = i
End Sub

```

编写按钮“按值传递的增1操作”的 Click 过程代码如下：

```

Private Sub Command2_Click()
    Dim i As Integer
    i = Val(Text1.Text)
    AddByRef i
    Text3.Text = i
End Sub

```

6.3.3 传递数组

除了可以使用变量作为参数之外，还可以用数组作为参数。在传送数组时，除遵守参数传递的一般规则外，还应注意以下几点：

- 1) 每一个过程只能引用一个数组作为参数。
- 2) 作为参数的数组只能放在过程所有参数的最后面。
- 3) 数组参数只能按地址传递，在形式参数数组前不能用 ByVal 修饰。
- 4) 数组参数只能是一维的。
- 5) 数组作为形式参数时，将数组名写入形式参数列表中，并略去数组的上下界，但数组名后的括号不能省略；实际参数数组的数据类型必须和形式参数数组的类型一致，实际参数数组后面的括号可以省略，但为了便于阅读，建议一般不要省略。

【例 6-5】 建立一个应用程序：将一个数组中的所有元素都进行增1操作。

设计步骤如下：

- 1) 建立应用程序用户界面和添加控件，并设置每个对象的属性，如图 6-10 所示。
- 2) 编写事件代码。编写通用过程 AddValue 代码，此过程实现参数数组的每一个元素值进行增1操作。代码如下：

```

Sub AddValue(b() As Variant)
    Dim i As Integer
    For i = 0 To UBound(b)
        b(i) = b(i) + 1
    Next i
End Sub

```

编写通用过程 ShowValue 代码，实现将数组中的值显示在标签控件中。代码如下：

```

Sub ShowValue(lb As Label, c() As Variant)
    Dim i As Integer
    Dim strTemp As String
    For i = 0 To UBound(c)
        If strTemp = "" Then
            strTemp = c(i)
        Else

```



```

        strTemp = strTemp & "," & c(i)
    End If
Next i
lb. Caption = strTemp
End Sub

```

3) 编写按钮“运行”的 Click 事件代码如下:

```

Private Sub Command1_Click()
    Dim a()
    Dim i As Integer
    a = Array(1, 2, 3, 4, 5, 6, 7, 8, 9)
    Call ShowValue( Label3, a())
    Call AddValue(a)
    Call ShowValue( Label4, a())
End Sub

```

4) 保存窗体，运行程序，结果如图 6-10 所示。

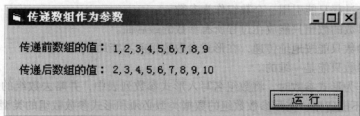


图 6-10 传递数组作为参数程序运行结果

6.4 过程的嵌套与递归调用

在一个过程中调用另一个过程，称为过程的嵌套调用；而过程直接或间接地调用其自身，称为过程的递归调用。

6.4.1 过程的嵌套调用

在 VB 中定义一个过程时，不能包括另一个过程的定义。但可以在一个过程中嵌套调用其他过程。调用其他过程的过程称为主过程，被调用的过程成为子过程。

过程嵌套调用的执行过程如图 6-11 所示。

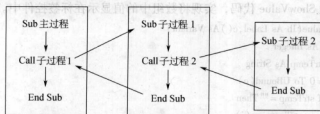


图 6-11 过程嵌套调用的执行过程

6.4.2 过程的递归调用

VB 的过程具有递归调用功能。例如，对阶乘的定义如下：

$$n! = n * (n-1) \\ (n-1)! = (n-1) * (n-2)$$

VB 允许在一个 Sub 子过程和 Function 过程的定义内部调用自己，即递归 Sub 子过程和递归 Function 函数。

【例 6-6】编写阶乘 $\text{fac}(n) = n!$ 的递归函数。

代码如下：

```
Private Function fac(n As Integer) As Integer
    If n = 1 Then
        fac = 1
    Else
        fac = n * fac(n-1)
    End If
End Function
```

6.5 应用举例

下面是通用过程解决实际问题的几个应用。

6.5.1 查找问题

查找是在已知的数据中，查找一个指定的数据，或和已知数据相关的数据。

【例 6-7】设计一个窗体，使用顺序查找法，输入数组中任何一个数便可以在数组中指定数的位置。

算法设计：

设一组数据存放在数组 $a(1) \cdots a(n)$ 中，待查找的数据放在 x 中，把 x 与 a 数组中的元素从头到尾一一进行比较查找。用变量 p 表示 a 数组元素下标， p 初值为 1，使 x 与 $a(p)$ 比较，如果 x 不等于 $a(p)$ ，则使 $p = p + 1$ ，不断重复这个过程；一旦 x 等于 $a(p)$ 则退出循环；另外，如果 p 大于数组长度，循环也应该停止。

操作步骤如下：

- 1) 窗体和控件的属性如图 6-12 所示。
- 2) 在代码设计窗口，定义一个 Function 过程 Find，用来实现在指定的数组中查找某个值，若找到则返回此值在数组中的下标的位置，否则返回值 -1。Find 过程的代码如下：

```
Function Find(a() As Variant, x As Integer) As Integer
    Dim i As Integer
    For i = 0 To UBound(a)
        If a(i) = x Then
            Exit For
        End If
    
```



```

Next i
If i > UBound(a) Then
    i = -1
End If
Find = i
End Function

```

3) 编写按钮“查找”的 Click 事件代码如下:

```

Private Sub Command1_Click()
    Dim data()
    Dim d As Integer
    Dim p As Integer
    data = Array(12, 34, 45, 22, 20, 10, 23, 100)
    Label2.Caption = ""
    For i = 0 To UBound(data)
        Label2.Caption = Label2.Caption & data(i) & Space(2)
    Next i
    d = Val(Text1.Text)
    p = Find(data, d)
    Label4.Caption = "数据" & d & "在数组中的位置为:" & p
End Sub

```

4) 保存窗体, 运行程序, 结果如图 6-12 所示。

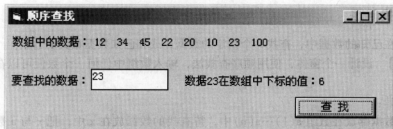


图 6-12 顺序查找程序运行结果

【例 6-8】 使用折半查找法, 在指定的有序数列中查找给定的数 x_0 。

折半查找算法分析:

设 n 个有序数 (从小到大) 存放在数组 $a(1) \cdots a(n)$ 中, 要查找的数为 x_0 。用变量 $start$ 、 $tail$ 、 mid 分别表示查找数据范围的底部 (数组下界)、顶部 (数组的上界) 和中间, $mid = (start + tail) / 2$, 折半查找的算法如下:

- 1) $x = a(mid)$, 则已找到退出循环, 否则进行下面的判断。
- 2) $x < a(mid)$, x 必定落在 $start$ 和 $mid - 1$ 的范围之内, 即 $tail = mid - 1$ 。
- 3) $x > a(mid)$, x 必定落在 $mid + 1$ 和 $tail$ 的范围之内, 即 $start = mid + 1$ 。
- 4) 在确定了新的查找范围后, 重复进行以上比较, 直到找到或者 $start \leq tail$ 。

按上面的算法编写函数过程 $HalfFind$, 若找到则返回该数所在的下标值, 没找到则返回 -1 。程序代码如下:

```

Function HalfFind(a() As Variant, x As Integer) As Integer
    Dim boolFind As Boolean
    Dim start, mid, tail As Integer
    start = LBound(a)
    tail = UBound(a)
    boolFind = False '判断是否找到的逻辑变量
    Do While start <= tail And boolFind = False
        mid = (start + tail) \ 2
        If a(mid) = x Then
            boolFind = True
            Exit Do
        ElseIf x < a(mid) Then
            tail = mid - 1 '查找上半部分
        Else
            start = mid + 1 '查找下半部分
        End If
    Loop
    If boolFind = True Then
        HalfFind = mid '若找到返回数组的元素的下标
    Else
        HalfFind = -1 '未找到返回值-1
    End If
End Function

```

读者可自己编写程序调用上面的函数，进行验证。

6.5.2 插入问题

插入是指在一个有序的数据序列中，插入一个数据后，仍然保持数据序列的有序性。

【例 6-9】 在指定的有序数列中插入给定的数 x 。

算法分析：

设 n 个有序数据(从小到大)存放在数组 $a(1) \cdots a(n)$ 中，要插入的数为 x 。首先将数组元素个数增大；然后确定 x 插在数组中的位置 p ，并将位置 p 及之后的数组元素向后移动；最后将 x 插入到位置 p 上。

操作步骤如下：

- 1) 窗体和控件的属性如图 6-13 所示。
- 2) 在代码设计窗口，定义一个 Sub 过程 Insert，用来实现在指定的有序数组 $a()$ 中插入数值 x ，插入后仍然保持数组 a 的有序性，Insert 过程的代码如下：

```

Private Sub insert(a() As Variant, x As Integer)
    Dim p, n, i As Integer
    n = UBound(a)
    ReDim Preserve a(n + 1) '让数组长度增加 1，以便存放插入的数
    p = LBound(a)

```



Do While x > a(p) And p <= n '确定 x 应插入的位置

p = p + 1

Loop

For i = n To p Step - 1 '向后移动数据

a(i + 1) = a(i)

Next i

a(p) = x

End Sub

3) 编写按钮“插入”的 Click 事件代码如下:

Private Sub Command1_Click()

Dim data()

Dim d As Integer

Dim p As Integer

data = Array(12, 34, 45, 52, 67, 88, 90, 100)

Label3.Caption = ""

Label4.Caption = ""

For i = 0 To UBound(data) '显示插入前的数据

Label3.Caption = Label3.Caption & data(i) & Space(2)

Next i

d = Val(Text1.Text)

Call insert(data, d)

For i = 0 To UBound(data) '显示插入后的数据

Label4.Caption = Label4.Caption & data(i) & Space(2)

Next i

End Sub

4) 保存窗体, 运行程序, 结果如图 6-13 所示。

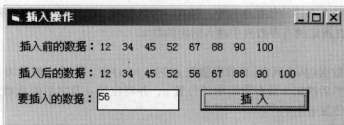


图 6-13 插入操作程序运行结果

习 题

- 6-1 Sub 过程和 Function 过程有什么区别?
- 6-2 形式参数和实际参数有什么区别?
- 6-3 形式参数和实际参数的传递方式有哪些? 有什么不同?
- 6-4 Private 和 Public 定义的 Sub 过程有什么不同?

6-5 编写程序, 计算表达式 $\frac{m!}{n! + (m-n)!}$ 的值 ($m > n \geq 0$), 要求: 用对话框输入 m 和 n 的值, 用编写函数来计算某个数的阶乘, 通过调用此函数来计算此表达式的值。

6-6 编写程序, 计算表达式 $e = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!}$ 的值, 并在窗体上输出。其中: x 和 n 的值由窗体上的文本框输入。

6-7 编写一个函数过程, 求出从整数 M 到整数 N 的和。

6-8 编写程序: 计算函数 $F = 1! + 2! + 3! + \cdots + (N+1)! + N!$ 的值。

6-9 程序如下, 写出程序的运行结果。

```
Public Function f(ByVal n as Integer, ByVal r As Integer)
```

```
    If n <= 0 then
```

```
        f(n\r, r)
```

```
    Print n mod r;
```

```
    End If
```

```
End Function
```

```
Private sub Command_Click()
```

```
    Print f(100, 8)
```

```
End sub
```

6-10 程序如下, 写出程序的运行结果。

```
Public Sub Proc(a() As Integer)
```

```
    Dim i As Integer
```

```
    Do
```

```
        a(i) = a(i) + a(i+1)
```

```
        i = i + 1
```

```
    Loop While i < 3
```

```
End Sub
```

```
Private Sub Command1_Click()
```

```
    Dim m, i As Integer
```

```
    Dim x(10) As Integer
```

```
    For i = 0 to 4
```

```
        x(i) = i + 1
```

```
    Next i
```

```
    Call Proc(x)
```

```
    For i = 0 to 4
```

```
        Print x(i)
```

```
    Next i
```

```
End Sub
```

变量	初始值	操作	结果
变量	1	1+2	3
变量	2	2+3	5
变量	3	3+5	8
变量	4	4+8	12
变量	5	5+12	17

第7章 绘 图

图形是 Windows 应用程序的重要应用领域,可以为应用程序的界面增添良好的视觉效果,提供可视化的结构。VB 提供了非常丰富的绘图功能,利用这些功能,可以为应用程序的界面增加吸引力。程序设计时,可以使用 VB 提供的图形控件画图,也可以调用图形方法绘制丰富多彩的艺术图形和直观统计图表。

7.1 绘图的相关知识

7.1.1 坐标系

坐标系是绘图的基础,在了解绘图之前,要先了解坐标系。实际上,VB 在很多地方都要用到坐标系,每个对象定位于存放它的容器内,对象定位都要使用容器的坐标。在向工程添加控件时,控件的属性设置中的 Top 属性和 Left 属性就是给控件定位的,通过改变这两个属性值,就可以确定控件的位置,这里用到的就是坐标系。

坐标系是一个二维的网格,用于定义容器对象(如窗体和图片框)中点的位置。与平面几何中的坐标类似,它由两个轴组成,沿着水平的方向称为 X 轴,沿着垂直的方向称为 Y 轴,两条轴相交的地方称为原点。要表示一个点在坐标中的位置用(x,y),其中 x 表示这个点在 X 轴方向上的位置,y 表示此点在 Y 轴方向上的位置,因此(0,0)表示坐标中两轴的交叉点。与平面几何中的坐标不同的是,VB 中坐标系的默认坐标原点(0,0)在容器对象的左上角,水平向右延伸为 X 轴正方向,垂直向下延伸为 Y 轴正方向,如图 7-1 所示。

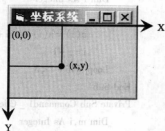


图 7-1 VB 坐标系

1. 刻度单位

VB 共提供了 8 个坐标系刻度单位,默认坐标使用单位缃(Twip)。程序设计者可以根据实际需要通过对 ScaleMode 属性的值来改变刻度单位。ScaleMode 属性取值见表 7-1。

表 7-1 ScaleMode 属性取值

属性值	说 明	属性值	说 明
0—User	用户自定义,可设置 ScaleLeft、ScaleTop、ScaleWidth、ScaleHeight 属性	4—Character	字符
1—Twip	缃(默认值),1440 缃等于 1 英寸	5—Inch	英寸
2—Point	点,72 点等于 1 英寸	6—Millimeter	毫米
3—Pixel	像素,表示分辨率的最小单位	7—Centimeter	厘米

ScaleMode 属性值 1~7 对应的坐标系中, X 轴正方向均水平向右, Y 轴正方向均垂直向下, 区别只是坐标系的刻度单位不同, 这 7 种坐标系是 VB 提供的标准坐标系。ScaleMode 属性值为 1 对应的坐标系是默认坐标系。

例如:

Form1.ScaleMode = 6 '设置窗体 Form1 的坐标系刻度单位为毫米

Picture2.ScaleMode = 3 '设置图片框 Picture2 的坐标系刻度单位为像素

2. 自定义坐标系

VB 中允许自定义坐标系, 自定义坐标系允许程序设计者灵活地定义原点位置、X 和 Y 轴方向和刻度。

(1) 通过 ScaleLeft、ScaleTop、ScaleWidth、ScaleHeight 属性自定义坐标系

当容器对象的 ScaleMode 属性设置为 0 时, 程序设计者可以利用容器对象相应的 4 个属性 ScaleLeft、ScaleTop、ScaleWidth、ScaleHeight 来定义自己认为合适的容器对象的坐标原点和刻度。其中:

ScaleLeft、ScaleTop 属性: 用于定义容器对象的左上角在新的自定义坐标系中的坐标值。

ScaleWidth、ScaleHeight 属性: 用于定义容器对象在新的自定义坐标系中的宽度和高度以及 X 轴和 Y 轴的正方向。若 ScaleWidth 的值为正数, 则 X 轴正方向水平向右, 若 ScaleWidth 的值为负数, 则 X 轴正方向水平向左; 若 ScaleHeight 的值为正数, 则 Y 轴正方向垂直向下, 若 ScaleHeight 的值为负数, 则 Y 轴正方向垂直向上。

当改变容器对象以上属性值时, 容器对象的左上角在新的自定义坐标系中的坐标值为 (ScaleLeft, ScaleTop), 容器对象的右下角在新的自定义坐标系中的坐标值为 (ScaleLeft + ScaleWidth, ScaleTop + ScaleHeight), 根据左上角和右下角坐标值的大小自动设置坐标轴的正方向。X 轴和 Y 轴的度量单位分别为 $1/\text{ScaleWidth}$ 和 $1/\text{ScaleHeight}$ 。

自定义坐标系 4 个属性的设置情况如图 7-2 所示。

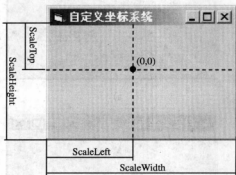
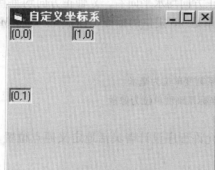


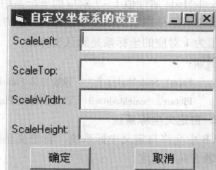
图 7-2 自定义坐标系 4 个属性的设置情况

从图 7-2 中可以看出, 自定义坐标系中的 4 个属性值是不包含标题栏和边框的。实际上, 所有以 Scale 为前导的属性名, 都是针对对象内的有效绘图区域的。

【例 7-1】 测试通过 ScaleLeft、ScaleTop、ScaleWidth、ScaleHeight 属性自定义坐标系。建立两个窗体 Form1、Form2, 如图 7-3a、b 所示。各控件的属性设置如表 7-2 所示。运行程序, 在 Form1 上单击鼠标左键弹出 Form2, 在 Form2 中对 ScaleLeft、ScaleTop、ScaleWidth、ScaleHeight 属性进行设置后, 单击“确定”按钮返回 Form1。在 Form1 中单击鼠标右键, 则 Form1 窗体中分别代表 (0,0)、(0,1)、(1,0) 3 个坐标点的 3 个标签 Label1、Label2、Label3 将移动到其在新的自定义坐标系中的相应的坐标位置(标签左上角的定位点对应相应的坐标点)。分别用表 7-3 左侧所示数值对以上 4 个属性进行设置, 则得到如图 7-4a~d 所示结果, 其分析见表 7-3 右侧。



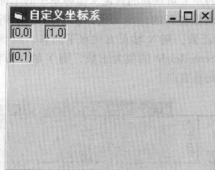
a)



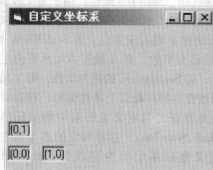
b)

图 7-3 程序界面

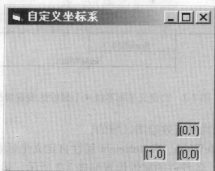
a) Form1 窗体 b) Form2 窗体



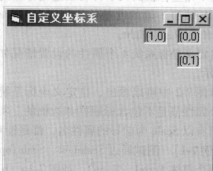
a)



b)



c)



d)

图 7-4 程序运行结果

a) X 轴向右, Y 轴向下 b) X 轴向右, Y 轴向上
c) X 轴向左, Y 轴向上 d) X 轴向左, Y 轴向下

表 7-2 各控件的属性设置

控件名称	属性名	属性值	控件名称	属性名	属性值
Form1	Caption	自定义坐标系	Form2	Caption	自定义坐标系的设置
	Height	2745		Height	2745
	Width	3420		Width	3420
Form1. Label1	AutoSize	True	Form2. Label1	Caption	ScaleLeft;
	BoderStyle	1—Fixed Single	Form2. Text1	Text	空
	Caption	(0,0)	Form2. Label2	Caption	ScaleTop;
Form1. Label2	AutoSize	True	Form2. Text2	Text	空
	BoderStyle	1—Fixed Single	Form2. Label3	Caption	ScaleWidth;
	Caption	(0,1)	Form2. Text3	Text	空
Form1. Label3	AutoSize	True	Form2. Label4	Caption	ScaleHeight;
	BoderStyle	1—Fixed Single	Form2. Text4	Text	空
	Caption	(1,0)	Form2. Command1	Caption	确定
			Form2. Command2	Caption	取消

程序代码如下：

'以下为 Form1 程序代码

```
Private Sub Form1_MouseDown(Button As Integer, _
```

```
Shift As Integer, X As Single, Y As Single)
```

```
Select Case Button
```

```
Case 1
```

```
Form2. Show 1, Form1
```

```
Case 2
```

```
Label1. Move 0,0
```

```
Label2. Move 0,1
```

```
Label3. Move 1,0
```

```
End Select
```

```
End Sub
```

'以下为 Form2 程序代码

```
Private Sub Command1_Click()
```

```
Form1. ScaleMode = 0
```

```
Form1. ScaleLeft = Val(Text1. Text)
```

```
Form1. ScaleTop = Val(Text2. Text)
```

```
Form1. ScaleWidth = Val(Text3. Text)
```

```
Form1. ScaleHeight = Val(Text4. Text)
```

```
Form2. Hide
```

```
End Sub
```

```
Private Sub Command2_Click()
```

```
Me. Hide
```



End Sub

表 7-3 自定义坐标系设置数据及分析

ScaleLeft	ScaleTop	ScaleWidth	ScaleHeight	X 轴正方向	Y 轴正方向	如 图
0	0	6	6	右	下	图 7-4a
0	5	6	-6	右	上	图 7-4b
5	5	-6	-6	左	上	图 7-4c
5	0	-6	6	左	下	图 7-4d

由表 7-3 可知, ScaleWidth 和 ScaleHeight 的值, 不仅决定自定义坐标系中宽度和高度的刻度值大小, 而且决定自定义坐标系中坐标轴的正方向。即当它们的值大于 0 时, 对应的坐标轴的正方向与标准坐标系统的坐标轴的正方向相同, 当它们的值小于 0 时, 对应的坐标轴的正方向与标准坐标系统的坐标轴的正方向相反。

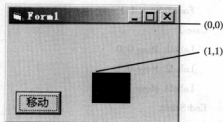
自定义的坐标值是相对的而且是没有量纲的值。例如, 某容器对象的 ScaleWidth = 10, 假设由于某种原因, 该对象的实际宽度增大了一倍, 但它的 ScaleWidth 仍为 10。即一旦 ScaleWidth、ScaleHeight 属性值被设定, 则不论容器对象的实际大小如何变化, 坐标的刻度值是不变的。利用这一特点常常可以使程序设计得到简化。

【例 7-2】 将一个标签 (Label1) 的左上角移动到窗体的中央位置。

如果采用默认坐标系, 就需要以缇为单位来计算窗体中央的位置, 显然计算略显繁琐。如果采用自定义坐标系, 将窗体的高度和宽度均定义为 2 个刻度单位, 则窗体的中央位置即是坐标为 (1,1) 的点。界面设计如图 7-5a 所示, 为便于区分, 将 Label1 的 BackColor 属性设置为黑色。运行时单击“移动”按钮, 将标签 Label1 的左上角移动到窗体的中央位置, 如图 7-5b 所示。



a)



b)

图 7-5 自定义坐标系的应用

a) 界面设计 b) 运行结果

“移动”按钮 Command1 的 Click 事件过程如下:

```
Private Sub Command1_Click()  
    Form1.ScaleWidth = 2  
    Form1.ScaleHeight = 2  
    Label1.Left = 1  
    Label1.Top = 1
```

End Sub

(2) 通过 Scale 方法自定义坐标系

Scale 方法可以重新设置各种容器对象的坐标系。其使用格式如下：

[<对象名>]. Scale[(x1,y1) - (x2,y2)]

该方法用于将容器对象的左上角坐标定义为(x1,y1)，右下角坐标定义为(x2,y2)。如果不带任何参数调用该方法，可以使坐标系还原成系统默认的坐标系。

一旦执行了 Scale 方法，该容器对象的 ScaleMode 值会自动变为 0，表示自定义坐标系，其他 4 个属性(ScaleLeft、ScaleTop、ScaleWidth、ScaleHeight)的值也将自动被设置成与 x1、y1、x2、y2 相对应的值。其中，x1、y1 的值决定了 ScaleLeft 和 ScaleTop 属性的值，而(x1,y1)与(x2,y2)两点 x 坐标的差值和 y 坐标的差值，分别决定了 ScaleWidth 和 ScaleHeight 属性的值。

例如，Picture1.Scale(10,20) - (200,300)用于将图片框的左上角的坐标定义为(10,20)，右下角的坐标定义为(200,300)。

7.1.2 与绘图相关的属性

1. 当前坐标

当在容器中绘制图形或输出结果时，经常要将它们定位在某一希望的位置，这就必须获得某一点的坐标，即当前坐标。属性 CurrentX 和 CurrentY 用于设置或返回当前坐标的水平坐标和垂直坐标。

例如，在点(300,150)处显示“当前坐标用于输出定位”，可以使用以下语句：

```
Form1.CurrentX = 300
```

```
Form1.CurrentY = 150
```

```
Form1.Print "当前坐标用于输出定位"
```

2. 线型样式

(1) DrawStyle 属性

该属性决定绘图方法输出时的线型样式。DrawStyle 属性的值见表 7-4。

表 7-4 DrawStyle 属性的值

属性值	常量	线型	属性值	常量	线型
0	vbSolid	实线(默认)	4	vbDashDotDot	双点画线
1	vbDash	虚线	5	vbInvisible	透明线(无线)
2	vbDot	点线	6	vbInsideSolid	内实线
3	vbDashDot	点画线			

(2) BorderStyle 属性

该属性用于使用控件时给出画线的样式。BorderStyle 属性的值见表 7-5。

表 7-5 BorderStyle 属性的值

属性值	常量	线型	属性值	常量	线型
0	vbTransparent	透明线(无线)	4	vbBSDashDot	点画线
1	vbBSSolid	实线(默认)	5	vbBSDashDotDot	双点画线
2	vbBSDash	虚线	6	vbBSInsideSolid	内实线
3	vbBSDot	点线			



表 7-4、表 7-5 中各种线型如图 7-6 所示。

3. 线宽

(1) DrawWidth 属性

该属性决定绘图方法输出时的线宽。线宽的取值范围为 1 ~ 32767，以像素为单位，默认值为 1，即一个像素宽。

当 DrawWidth 属性值大于 1 并且 DrawStyle 属性值为 1 ~ 4 时，都能产生实线效果。

当 DrawWidth 属性值等于 1 时可以画出 DrawStyle 属性值决定的各种线型的图形。

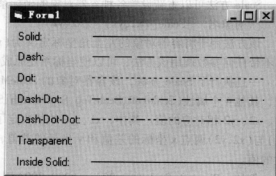


图 7-6 线型

(2) BorderWidth 属性

该属性用于使用控件时定义线的宽度。

当 BorderWidth 属性值大于 1 并且 BorderStyle 属性值为 2 ~ 5 时，都能产生实线效果。

当 BorderWidth 属性值等于 1 时可以画出 BorderStyle 属性值决定的各种线型的图形。

4. 填充

封闭图形的填充方式是由 FillColor 属性与 FillStyle 属性决定的。

(1) FillColor 属性

该属性指定填充图案的颜色，默认颜色与 ForeColor 相同。

(2) FillStyle 属性

该属性决定填充封闭图形的图案样式。FillStyle 属性的值见表 7-6，相应的填充样式如图 7-7 所示。

表 7-6 FillStyle 属性的值

属性值	常量	说明	属性值	常量	说明
0	vbFSSolid	实心	4	vbUpwardDiagonal	上斜对角线
1	vbFSTransparent	透明(默认)	5	vbDownwardDiagonal	下斜对角线
2	vbHorizontalLine	水平线	6	vbCross	交叉线
3	vbVerticalLine	垂直线	7	vbDiagonalCross	对角交叉线

如果 FillStyle 设置为 1 (透明)，则忽略 FillColor 属性。

5. 绘图方式

绘图方式直接影响绘图的输出结果。具体地说，它决定了在绘图区域图形输出时与原位置上的图形相重叠后的外观形象及颜色变化。属性 DrawMode 用于设置绘图方式，DrawMode 共有 16 种不同的属性值，见表 7-7。它们反映了对画笔前景颜色与当前显示的背景色所进行的各种布尔

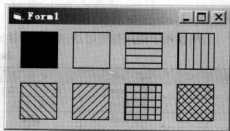


图 7-7 填充样式

代数运算。默认值为 13，表示直接用前景色覆盖背景色。

当使用 PSet、Line、Circle 等方法来绘制图形时，通过对 DrawMode 属性的设置，可绘制出丰富多彩的图形或动画。

表 7-7 DrawMode 属性的值

属 性 值	常 量	说 明
1	vbBlackness	用黑色显示输出图形
2	vbNotMergePen	Merge Pen(15)后的反相颜色
3	vbMaskNotPen	将当前显示色与画笔的反相颜色进行“与”混合
4	vbNotCopyPen	以画笔的反相颜色显示输出
5	vbMaskPenNot	将当前显示颜色的反相颜色和画笔颜色进行“与”混合
6	vbInvert	以当前显示的颜色的反相颜色显示输出
7	vbXorPen	将画笔的颜色与当前显示的颜色进行“异或”混合
8	vbNotMaskPen	Mask Pen(9)的反相颜色
9	vbMaskPen	将画笔颜色与当前显示颜色进行“与”混合
10	vbNotXorPen	以 Xor Pen(7)的反相颜色显示输出
11	vbNop	停止绘图操作，显示的图形保持原样
12	vbMergeNotPen	将画笔的反相色和当前显示色进行“与”混合
13	vbCopyPen	直接以画笔颜色(ForeColor)显示输出(默认设置值)
14	vbMergePenNot	将画笔颜色和当前显示色的反相色进行混合
15	vbMergePen	将当前显示色和画笔的颜色进行混合
16	vbWhiteness	用白色显示输出

6. 重新绘制

(1) AutoReDraw 属性

Windows 是一个多任务操作系统，应用程序在运行时其窗体经常被移动或被其他窗体覆盖。如果想保持窗体中的内容(图形等)不丢失，就要在窗体移动、改变大小或覆盖它的窗体移开后，重新显示(绘制)窗体中的内容。通常，Windows 管理和控制窗口及控件的重新显示，而窗体和图片框内图形的重新显示必须由用户的应用程序来控制。

利用 AutoReDraw 属性就可以实现对窗体和图片框内图形的重新显示的控制。当 AutoReDraw 属性为 False(默认值)时，对象中的图形不具有持久性，即当覆盖对象的窗体或控件被移动或改变大小后，对象上的图形将丢失；当 AutoReDraw 属性为 True 时，对象的自动重绘功能有效，图形具有持久性，系统会将以前自动保存在内存中的图形调出来重绘，使对象内的图形保持原有的样子。

为了节省内存，可以将窗体的 AutoReDraw 属性设置为 False，但那样的图形将不再自动成为持久的，需要时必须使用代码来管理所有图形的重绘。

(2) ClipControls 属性

ClipControls 属性用于决定 Paint 事件中的图形方法是对整个对象还是对刚刚露出的区域进行重绘。它还可以决定 Windows 运行环境是否创建一个不包括该对象的非图形控件的剪

裁区。

该属性是一个逻辑值,在运行时为只读状态。当 ClipControls 属性为 True(默认值)时,Paint 事件中的图形方法重绘整个对象,在绘制之前,在该窗体的非图形控件的周围创建剪裁区。当 ClipControls 属性为 False 时,Paint 事件中的图形方法只绘制刚刚露出的区域,在绘制之前,不在该窗体非图形区域的周围创建剪裁区。此时加载和重绘复杂窗体比较快。

剪裁区包括大部分控件,但不包括图形控件(Image、Line、Shape)和 Label 控件。注意:应尽量避免 ClipControls 属性为 True 的控件嵌入到 ClipControls 属性为 False 的容器中,这样会导致嵌入的控件不能被正确地重绘。

7.1.3 颜色

VB 中经常要涉及颜色,如设置字体的颜色、填充的颜色、绘图线条的颜色以及控制对象的前景、背景颜色等。在 VB 中,颜色值是一个 4 字节的长整型(Long)数,其中最低的 3 个字节分别对应于构成颜色的三原色:红、绿、蓝。每个字节的取值范围以十进制数表示为 0~255,故 3 个字节组合在一起,可表示 2^{24} (16777216) 种颜色(实际显示的颜色与显卡和显示器有关)。在 VB 中,设置或获取颜色值有多种实现方法。

1. 使用调色板对话框

调色板对话框是在程序设计阶段设置对象颜色的简单、易行的手段,通过调色板对话框可以直观、可视地设置当前对象的颜色。

调色板对话框的打开,有两种方式。方式 1 是从对象属性窗口中选择颜色属性(如 ForeColor、BackColor),单击右面的下拉箭头,打开如图 7-8 所示的“调色板”对话框,其中包含两个选项卡,一个显示的是含有 48 种颜色的调色板,如图 7-8a 所示,另一个显示的是系统预定义的颜色,如图 7-8b 所示。使用时可从两个选项卡中任选其一,再从中设置对象的颜色。

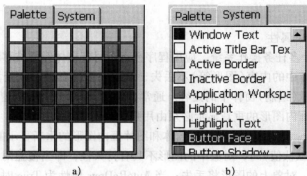


图 7-8 通过对象颜色属性窗口打开“调色板”对话框

a) 调色板 b) 系统颜色

采用方式 1 打开的“调色板”对话框,其配色方案是固定的。如果希望在调色板中增加若干经常使用的其他颜色,并能长期方便地使用,则应采用方式 2 打开“调色板”对话框。方式 2 是通过单击“视图”下拉菜单下的“调色板”菜单项,打开如图 7-9a 所示的“调色板”对话框。单击左上方小方块的中央,然后单击右侧合适的颜色格,是设置选中对象的前景色;单击左上方小方块的周围部分,然后单击右侧合适的颜色格,是设置选中对象

的背景色。左下方小方块为颜色设置的实时预览区，其中的字符“Aa”代表前景色，其余部分代表背景色。单击最下面一行 16 个小方格中的任何一个，然后通过单击“Define Colors...”按钮弹出如图 7-9b 所示的自定义颜色对话框，可自定义在调色板中没有的新的配色方案。单击“Default”按钮将恢复到默认的颜色设置。

2. 使用 ShowColor 方法

如果在程序运行阶段需要设置对象的颜色，可以在程序代码中利用通用对话框控件(Common Dialog)的 ShowColor 方法打开一个标准的“颜色”对话框，如图 7-10 所示，通过选择来确定所需要的颜色值。

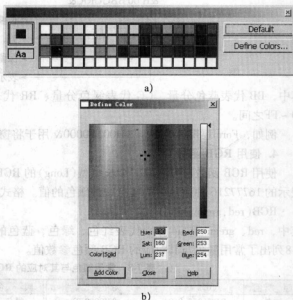


图 7-9 通过下拉菜单打开“调色板”对话框

a) 可自定义颜色的调色板 b) 自定义颜色对话框

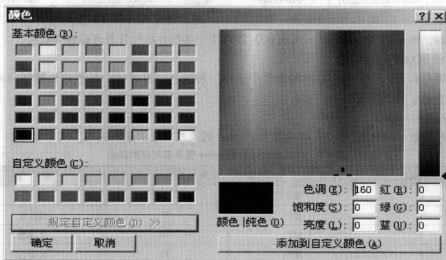


图 7-10 使用通用对话框控件打开“颜色”对话框

3. 使用颜色的十六进制数表示

VB 内部使用十六进制数代表指定的颜色。如果在程序运行阶段需要设置对象的颜色，可以在程序代码中直接使用该十六进制数为颜色属性赋值；在程序设计阶段也可在对象的颜色属性值中直接输入该十六进制数进行设置。

该十六进制数表示如下：

&H 00 BBGGRR &

长整型符号

蓝、绿、红三原色

保留

表示十六进制数

其中, BB 代表蓝色分量、GG 代表绿色分量、RR 代表红色分量, 它们的取值范围均介于 00 ~ FF 之间。

例如, Form1.BackColor = &H00FF0000& 用于将窗体的背景色设置为蓝色。

4. 使用 RGB 函数

使用 RGB 函数可以获取一个长整型(Long)的 RGB 颜色值, 通过该函数可以得到 VB 可表示的 16777216 种颜色中的任意一种颜色的值。格式如下:

RGB(red, green, blue)

其中, red、green、blue 分别代表红色、绿色、蓝色的值, 取值范围是 0 ~ 255 的整数。表 7-8 列出了常用颜色与其对应的 RGB 颜色参数值。

表 7-8 常用颜色与其对应的 RGB 颜色参数值

颜色	RGB 函数	对应颜色值	颜色	RGB 函数	对应颜色值
红色	RGB(255,0,0)	&H0000FF	黑色	RGB(0,0,0)	&H000000
绿色	RGB(0,255,0)	&H00FF00	黄色	RGB(255,255,0)	&H00FFFF
蓝色	RGB(0,0,255)	&HFF0000	紫红色	RGB(255,0,255)	&HFF00FF
白色	RGB(255,255,255)	&HFFFFFF	青蓝色	RGB(0,255,255)	&HFFFF00

例如, Picture1.BackColor = RGB(0,0,255) 用于将图片框的背景色设置为红色。

5. 使用 QBColor 函数

使用 QBColor 函数可以从 16 种颜色中选择一种颜色, 返回选中颜色的长整型(Long)的 RGB 颜色值。格式如下:

QBColor(value)

颜色参数 value 是介于 0 ~ 15 的整数, value 值及其对应的颜色见表 7-9。

表 7-9 QBColor 函数的 value 值及其对应的颜色

参数值(value)	颜色	参数值(value)	颜色
0	黑色	8	灰色
1	蓝色	9	亮蓝色
2	绿色	10	亮绿色
3	青色	11	亮青色
4	红色	12	亮红色
5	品红色	13	亮品红色
6	黄色	14	亮黄色
7	白色	15	亮白色

例如, Form1.BackColor = QBColor(2) 用于将窗体的背景色设置为绿色。

6. 使用颜色常量

为方便用户使用, VB 将经常使用的颜色值定义为内部颜色常量。内部颜色常量可以直接被程序引用, 而无须重新定义。VB 定义的颜色常量如表 7-10 所示。

表 7-10 颜色常量

颜色常量	颜色	颜色常量	颜色
vbBlack	黑色	vbMagenta	品红色
vbBlue	蓝色	vbRed	红色
vbCyan	青色	vbWhite	白色
vbGreen	绿色	vbYellow	黄色

例如, `Picture1.BackColor = vbCyan` 用于将图片框的背景色设置为青色。

7. 使用系统颜色

系统颜色是操作系统本身定义的颜色, VB 允许在应用程序中直接引用系统颜色设置窗体和控件的颜色属性。其优点是保持应用程序与系统的一致性, 当用户在 Windows 的控制面板中改变了系统颜色对应的颜色值, 应用程序中被引用的相应颜色也会随之变化。

系统颜色在 VB 中有两种表示方法, 一种是用十六进制数表示的 4 字节长整数, 另一种是用内部常量。用十六进制数表示的系统颜色值的第一个字节为 80, 其余字节指定的是一种系统颜色。在图 7-8b 的属性窗口中, 选择一种系统颜色后, 可以看出所对应的颜色值。例如, 选择窗体文本 (Window Text) 颜色, 所产生的颜色值为 `&H80000008`, 内部常量为 `vbWindowText`; 而选择按钮文本 (Button Text) 颜色, 所产生的颜色值为 `&H80000012`, 内部常量为 `vbButtonText`。

7.2 绘图操作

7.2.1 图形控件

图形控件包括直线控件 (Line) 和形状控件 (Shape), 如图 7-11 所示, 是 VB 提供的两种标准绘图工具, 利用它们可在容器对象 (窗体、图片框) 上快速直接地绘制各种简单的线条及形状, 如直线、圆等。图形控件的属性, 既可以在设计阶段设置, 也可以在运行阶段由程序动态地改变。但图形控件无法响应事件过程, 因此主要用于在设计状态下对容器对象进行界面装饰设计。



图 7-11 图形控件

1. 直线控件

直线控件 (Line) 用于在容器对象中创建简单的线段, 它没有自己的特殊方法, 也不产生任何事件。设计和运行时可以通过改变相应属性的值来改变它的位置、颜色和宽度等。直线控件的主要属性及其说明见表 7-11。



表 7-11 直线控件的主要属性及其说明

属 性 值	说 明	属 性 值	说 明
X1、Y1	直线起点的坐标值	BorderWidth	直线的宽度
X2、Y2	直线终点的坐标值	BorderColor	直线的颜色
BorderStyle	直线的样式(见表 7-5、图 7-6)	DrawMode	绘图方式(见表 7-7)

当 BorderStyle 属性为“0”(透明)时,将忽略 BorderColor 属性的设置值。

直线控件不含 Left、Top、Width 和 Height 属性,运行时也不能用 Move 方法决定直线的位置和长度,其位置和长度的设置是通过它的特有属性 X1、Y1、X2、Y2 来实现的。X1、Y1 决定直线起点的坐标, X2、Y2 决定直线终点的坐标,通过改变 X1、Y1、X2、Y2 属性的值,可以改变直线的位置和长度。

例如,以下代码将在窗体的绘图区中间画一条垂直的直线,将窗体分割为左右相等的两部分:

```
Line1.X1 = Form1.ScaleWidth/2
```

```
Line1.Y1 = Form1.ScaleTop
```

```
Line1.X2 = Form1.ScaleWidth/2
```

```
Line1.Y2 = Form1.ScaleHeight
```

2. 形状控件

形状控件(Shape)用于在容器对象中绘制填充或不填充的矩形、正方形、椭圆、圆、圆角矩形以及圆角正方形。形状控件的主要属性及其说明见表 7-12。形状控件 Shape 属性的值见表 7-13。

表 7-12 形状控件的主要属性及其说明

属 性 值	说 明	属 性 值	说 明
Left、Top	左上角的水平位置和垂直位置	DrawMode	绘图方式(见表 7-7)
Width、Height	宽度和高度	FillStyle	填充样式(见表 7-6、图 7-7)
BorderStyle	边界线的样式(见表 7-5、图 7-6)	FillColor	填充颜色
BorderWidth	边界线的宽度	Shape	形状的设置(见表 7-13、图 7-12b)
BorderColor	边界线的颜色		

表 7-13 形状控件 Shape 属性的值

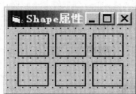
属 性 值	常 量	形 状	属 性 值	常 量	形 状
0(默认值)	vbShapeRectangle	矩形	3	vbShapeCircle	圆形
1	vbShapeSquare	正方形	4	vbShapeRoundedRectangle	圆角矩形
2	vbShapeOval	椭圆形	5	vbShapeRoundedSquare	圆角正方形

【例 7-3】在窗体中使用 Shape 控件,画 6 个不同形状的图形。

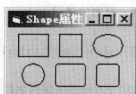
设计界面如图 7-12a 所示,在窗体上布置 6 个 Shape 控件,分别是 Shape1 ~ Shape6,运行时单击窗体,将图形设置为 6 种不同的形状,如图 7-12b 所示。窗体的 Click 事件过程如

下:

```
Private Sub Form_Click()  
    Shape1.Shape = vbShapeRectangle  
    Shape2.Shape = vbShapeSquare  
    Shape3.Shape = vbShapeOval  
    Shape4.Shape = vbShapeCircle  
    Shape5.Shape = vbShapeRoundedRectangle  
    Shape6.Shape = vbShapeRoundedSquare  
End Sub
```



a)



b)

图 7-12 形状控件的 Shape 属性

a) 设计界面 b) 运行界面

7.2.2 绘图方法

VB 除提供了两个图形控件外,还提供了多种绘图方法,因而可在容器对象上完成复杂图形的设计。但是,绘图方法只能出现在程序代码中,只有当应用程序运行时才能显示出用绘图方法所绘制的图形。

1. 画点方法 (PSet)

格式如下:

[<对象名>].PSet[Step](x,y)[,<颜色>]

功能:

PSet 方法用于在对象的指定位置,用指定的颜色画一个点。

说明:

- 1) <对象名>: 可选项,要绘制点的容器对象的名称,如窗体、图片框等,默认为当前窗体。
- 2) (x,y): 欲绘制点的坐标,可以是任何数值表达式。
- 3) <颜色>: 绘制点的颜色值。如果省略,默认颜色为容器对象的前景色 (ForeColor)。
- 4) Step: 可选项,带此参数时(x,y)是相对于当前坐标点 (CurrentX, CurrentY) 的坐标,否则为绝对坐标。
- 5) 点的大小由容器对象的 DrawWidth 属性决定。

【例 7-4】单击窗体时,用 PSet 方法在窗体上绘制一条 $[0^\circ, 360^\circ]$ 的余弦曲线。

为便于确定曲线中每一点的坐标,使用 Scale 方法定义窗体的水平坐标从左到右为 $0 \sim 360$,垂直坐标从下到上为 $-1 \sim 1$,即 Scale (0,1) ~ (360,-1)。运行时单击窗体,结果如图 7-13 所示。



窗体的 Click 事件过程如下:

```
Private Sub Form_ Click()
    Scale (0,1)-(360,-1)
    DrawWidth = 2
    For x = 0 To 360
        y = 0.9 * Cos(x * 3.1415926/180)
        PSet (x, y), vbBlue
    Next x
EndSub
```



图 7-13 余弦曲线的绘制运行结果

2. 画直线、矩形方法 (Line)

格式如下:

[<对象名>].Line[Step][(x1,y1)]-[Step](x2,y2)[,<颜色>][,<B[F]>]

功能:

Line 方法用于在绘图区域内的任意两点间画直线或矩形。

说明:

1) <对象名>: 可选项, 要绘制直线或矩形的容器对象的名称, 如窗体、图片框等, 默认为当前窗体。

2) (x1,y1): 可选项, 起点坐标。如果省略, 图形起始于当前坐标点 (CurrentX, CurrentY)。

3) (x2,y2): 终点坐标。

4) Step: 可选项, 当 (x1,y1) 之前带有 Step 时, 表示 (x1,y1) 是相对于由当前坐标点 (CurrentX, CurrentY) 指示的位置; 当 (x2,y2) 之前带有 Step 时, 表示 (x2,y2) 是相对于图形起点的终点坐标。

5) <颜色>: 绘制直线或矩形的颜色值。如果省略, 默认颜色为容器对象的前景色 (ForeColor)。

6) B: 可选项, 如果使用了参数 B, 则以 (x1,y1)、(x2,y2) 为对角坐标画出矩形。

7) F: 可选项, 只有使用了参数 B, 才可以使用参数 F。如果使用了参数 F, 则规定矩形以矩形边框的颜色填充; 如果不使用参数 F, 则矩形的填充由当前容器对象的 FillColor 和 FillStyle 属性 (见表 7-6) 决定。

8) 直线或矩形边框的线宽由容器对象的 DrawWidth 属性决定。

9) 画连续直线时, 前一条直线的终点就是后一条直线的起点。执行 Line 方法后, 当前坐标点 (CurrentX, CurrentY) 被设置在终点坐标 (x2,y2) 处。

【例 7-5】单击窗体时, 用 Line 方法在窗体上绘制如图 7-14 所示图形。

窗体的 Click 事件过程如下:

```
Private Sub Form_ Click()
    Scale(0,0)-(10,10)
    DrawWidth = 5
    画菱形
    Line(1,5)-Step(1,-2.5), vbRed
```

```
Line-Step(1,2.5), vbRed
Line-Step(-1,2.5), vbRed
Line-Step(-1,-2.5), vbRed
'画三角形
CurrentX = 5
CurrentY = 2.5
Line-Step(1.5,5), vbBlue
Line-Step(-3,0), vbBlue
Line-Step(1.5,-5), vbBlue
'画矩形
CurrentX = 7.5
current = 2.5
Line-Step(1.5,0), vbGreen
Line-Step(0,5), vbGreen
Line-Step(-1.5,0), vbGreen
Line-Step(0,-5), vbGreen
```

End Sub

3. 画圆方法 (Circle)

格式如下:

[<对象名>].Circle[Step][(x,y)],<半径>,[<颜色>],[<起始角>],[<终止角>][,<纵横比>]

功能:

Circle 方法用于在容器对象上画圆形、椭圆形、圆弧和扇形。

说明:

1) <对象名>: 可选项, 要绘制图形的容器对象的名称, 如窗体、图片框等, 默认为当前窗体。

2) Step: 可选项, 带此参数时(x,y)是相对于当前坐标点(CurrentX,CurrentY)的坐标, 否则为绝对坐标。

3) (x,y): 圆、椭圆、弧或扇形的圆心坐标。

4) <半径>: 圆、椭圆、弧或扇形的半径。若为椭圆, 则为最长轴的长度。

5) <颜色>: 可选项, 圆、椭圆、弧或扇形的边框颜色值。如果省略, 默认颜色为容器对象的前景色(ForeColor)。

6) <起始角>: 可选项, 指定弧的起点位置(以弧度为单位)。取值范围为 $-2\pi \sim 2\pi$, 默认值是0(水平轴的正方向)。若为负数, 则在画弧的同时还要画出圆心到弧的起点的连线。绘制起始角度从0开始的扇形时, 要赋给起始角一个很小的负数, 如 -0.00001 。

7) <终止角>: 可选项, 指定弧的终点位置(以弧度为单位)。取值范围为 $-2\pi \sim 2\pi$, 默认值是 2π (从水平轴的正方向逆时针旋转 360°)。若为负数, 则在画弧的同时还要画出圆心到弧的终点的连线。弧的画法是从起点逆时针画到终点。

8) <纵横比>: 可选项, 圆的纵轴和横轴的长度比, 默认值为1, 表示画一个标准圆。当纵横比大于1时, 椭圆的纵轴比横轴长; 当纵横比小于1时, 椭圆的纵轴比横轴短。



图 7-14 用 Line 方法在窗体上绘制图形

9) 除圆心坐标和半径外, 其余参数均可省略, 但若省略的是中间参数, 则逗号必须保留。

10) 执行 Circle 方法后, 当前坐标点 (CurrentX, CurrentY) 被设置成圆心的坐标值。

【例 7-6】单击窗体时, 用 Circle 方法在窗体上绘制如图 7-15 所示图形。

```
Private Sub Form_Click()
```

```
Const pi = 3.1415926
```

```
Scale(0,0)-(10,10)
```

```
DrawWidth = 2
```

```
Circle(3,3),1
```

```
Circle(3,7),1,vbRed,,,0.5
```

```
Circle(3,7),1,vbBlue,,,2
```

```
Circle(7,3),1,vbCyan,-0.75 * pi,-0.25 * pi
```

```
Circle(7,7),1,vbGreen,-0.25 * pi,-0.75 * pi
```

```
Circle(7,7),1,,1.25 * pi,1.75 * pi
```

```
End Sub
```

'画标准圆

'画红色椭圆(纵轴短,横轴长)

'画蓝色椭圆(纵轴长,横轴短)

'画青色扇区

'画绿色扇区

'画弧

【例 7-7】单击窗体时, 用 Circle 方法在窗体上绘制如图 7-16 所示同心圆。

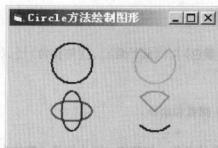


图 7-15 用 Circle 方法在窗体上绘制图形

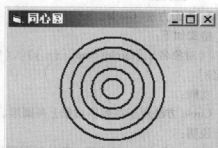


图 7-16 用 Circle 方法在窗体上绘制同心圆

```
Private Sub Form_Click()
```

```
Scale(0,0)-(10,10)
```

```
DrawWidth = 2
```

```
Circle(5,5),0.5
```

```
Circle(5,5),1
```

```
Circle(5,5),1.5
```

```
Circle(5,5),2
```

```
Circle(5,5),2.5
```

```
End Sub
```

4. 清除图形方法 (Cls)

格式如下:

[<对象名>].Cls

功能:

Cls 方法用于清除容器对象中生成的图形和文本, 将光标复位 (即移到原点)。

说明:

1) 〈对象名〉: 可选项, 要清除内容的容器对象的名称, 如窗体、图片框等, 默认为当前窗体。

2) 该方法执行后, 对象的 CurrentX、CurrentY 属性将被复位为 0。

5. 取点颜色方法(Point)

格式如下:

[〈对象名〉.]Point(x,y)

功能:

Point 方法用于获取容器对象上指定坐标点的颜色值。

说明:

1) 〈对象名〉: 可选项, 要获取颜色的容器对象的名称, 如窗体、图片框等, 默认为当前窗体。

2) (x,y): 用于指定容器对象上的坐标点, 以返回该点的颜色值。

例如, Picture1.Point(100,200) 用于返回图片框上坐标点(100,200)处的颜色值。

【例 7-8】单击窗体时, 随机改变窗体的背景色, 然后用 Point 方法获取窗体背景色的颜色值, 并将组成背景色的三原色的值求出、显示。

设计界面如图 7-17a 所示, 在窗体上布置 3 个 Label 控件, 分别是 Label1 ~ Label3, 并将 3 个 Label 控件的 Caption 属性置空, BackStyle 属性置 0(透明)。运行时单击窗体, 窗体背景色将随机改变, 并显示出组成背景色的三原色的值, 如图 7-17b 所示。

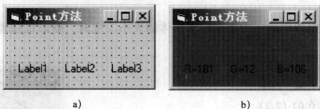


图 7-17 用 Point 方法获取颜色

a) 设计界面 b) 运行界面

窗体的 MouseUp 事件过程如下:

```
Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    '产生三原色的随机值
```

```
    r = 255 * Rnd
```

```
    g = 255 * Rnd
```

```
    b = 255 * Rnd
```

```
    '设置背景色
```

```
    BackColor = RGB(r, g, b)
```

```
    '获取相应点的颜色值
```

```
    c = Point(X, Y)
```

```
    '计算三原色中红色的值
```

```
    r = c Mod 256
```



```

'计算三原色中绿色的值
g = c \ 256 Mod 256
'计算三原色中蓝色的值
b = c \ 256 \ 256 Mod 256
'显示结果
Label1. Caption = "R =" & r
Label2. Caption = "G =" & g
Label3. Caption = "B =" & b

```

```
End Sub
```

6. Paint 事件

前面介绍了常用的绘图方法，用这些方法可以在容器对象上绘制各种各样的图形。但当容器对象被移动或改变大小之后，或当一个覆盖该容器对象的窗体被移开后，如果保持该对象上所画图形的完整性，则重现原来图形的工作应由用户自身负责（Windows 只负责容器对象本身的重现工作）。这时，可以选择 Paint 事件来完成图形的重画工作。

改变一个子控件（如用 Move 方法移动窗体上的 Label 控件）或改变某一属性（如改变 Picture 属性时），触发 Paint 事件。

使用 Refresh 方法时，触发 Paint 事件。

在改变窗体大小后，应在 Resize 事件过程中调用 Refresh 方法，强制对象通过 Paint 事件重画图形。

如果对象的 AutoRedraw 属性被设置为 True，重新绘图将自动进行，此时 Paint 事件无效。

【例 7-9】在窗体中画一个菱形，当窗体的大小改变时，菱形也随着自动调整，如图 7-18 所示。

程序代码如下：

```

Private Sub Form_Paint()
    Scale(0,0)-(2,2)
    Line(0,1)-(1,0)
    Line-(2,1)
    Line-(1,2)
    Line-(0,1)
End Sub

Private Sub Form_Resize()
    Refresh
End Sub

```



图 7-18 Paint 事件演示

7.3 键盘和鼠标操作

用户主要是通过键盘和鼠标与 VB 应用程序进行交互，所以应用程序必须可以响应键盘和鼠标事件。VB 应用程序能够响应多种键盘事件和鼠标事件，并且支持事件驱动的拖放功能。

7.3.1 键盘事件

键盘是用户与 VB 程序之间进行交互操作的主要工具之一。键盘事件是由键入产生的，与键盘有关的事件是 KeyDown 事件、KeyPress 事件和 KeyUp 事件。对于接收文本的控件，通常要对键盘事件编程。另外，还要提供既可用鼠标也可以用键盘操作的键盘事件代码，因为许多用户还需要用键盘工作，所以，考虑程序的键盘响应是很重要的。

1. KeyPress 事件

格式如下：

```
Private Sub <Object>_KeyPress([Index As Integer,]KeyAscii As Integer)
```

...

```
End Sub
```

说明：

- 1) 当按下下一个 ASCII 码键时，将引发 KeyPress 事件。VB 提供了 KeyPress 事件过程的框架，其格式如上所示。
- 2) <Object>：窗体或其他可以响应键盘事件的控件的名字。当该控件获得焦点时才能响应键盘事件。
- 3) Index：是一整型数，当 Object 是控件数组名时才有此参数。它是控件数组元素的下标，用于指定数组中一个控件成员。
- 4) KeyAscii：是被按按键对应字符的 ASCII 码值。每当键盘上的一个有 ASCII 码的键被按下时，该键的 ASCII 码就被存储到变量 KeyAscii 中，并引发 KeyPress 事件。其参数 KeyAscii 是传址调用，在 KeyPress 事件过程执行完以前，任何对象或过程都不能使用此变量。因此，可以利用 KeyPress 事件过程改变 KeyAscii 变量的值，使其他对象或过程接收到的并不是所敲键的 ASCII 码。根据这一原理，可以用它控制键盘输入字符的合法性。例如，以下代码将对输入文本框的字符进行检测，只准输入数字字符。

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
```

```
If KeyAscii < 48 Or KeyAscii > 57 Then KeyAscii = 0
```

```
'将 KeyAscii 置为 0 使控件看不到输入，即抑制了对
```

```
'控件的输入
```

```
End Sub
```

【例 7-10】如图 7-19 所示，每输入一个字符，将该字符及其对应的 ASCII 码值在窗体上显示出来。

窗体的 KeyPress 事件过程如下：

```
Private Sub Form_KeyPress(KeyAscii As Integer)
```

```
Print Chr(KeyAscii) & " : " & KeyAscii
```

```
End Sub
```

2. KeyDown 和 KeyUp 事件

格式如下：

```
Private Sub <Object>_KeyDown([Index As Integer,]Key-  
Code As Integer, Shift As Integer)
```

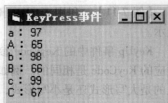


图 7-19 KeyPress 事件演示



.....

End Sub

Private Sub (Object) _KeyUp([Index As Integer,] KeyCode As Integer, Shift As Integer)

.....

End Sub

说明:

1) 当按下键盘上任一键时, 发生 KeyDown 事件; 放开按键时, 发生 KeyUp 事件。VB 提供了响应此两事件的事件过程的框架, 其格式如上所示。

2) (Object): 窗体或其他可以响应键盘事件的控件的名字。当该控件获得焦点时才能响应键盘事件。

3) Index: 是一整型数, 当 Object 是控件数组名时才有此参数。它是控件数组元素的下标, 用于指定数组中一个控件成员。

4) KeyCode: 按键的键盘码, 表示按键的物理位置。因此, 大写和小写字母将有相同的 KeyCode。

5) Shift: 该参数是一个 3 位二进制整数, 表示键盘事件发生时, 键盘上的 Shift、Ctrl 和 Alt 键是否被按下。其对应关系见表 7-14。这 3 键可以多选, 例如同时按下 Ctrl 和 Alt 键, 则 Shift 值为 110(二进制)。

表 7-14 Shift 参数的值

符号常数	二进制数	十进制数	含 义
vbShiftMask	001	1	Shift 键按下
vbCtrlMask	010	2	Ctrl 键按下
vbAltMask	100	4	Alt 键按下

表 7-14 中符号常数用作位屏蔽, 可被用来测试任何键组合。例如, 以下代码用于测试 Shift、Ctrl 和 Alt 三个键中哪两个被一起按下:

ShiftDown = (Shift And vbShiftMask) > 0

CtrlDown = (Shift And vbCtrlMask) > 0

AltDown = (Shift And vbAltMask) > 0

If (ShiftDown And CtrlDown) Then Print "Shift 键和 Ctrl 键被一起按下"

If (ShiftDown And AltDown) Then Print "Shift 键和 Alt 键被一起按下"

If (CtrlDown And AltDown) Then Print "Ctrl 键和 Alt 键被一起按下"

【例 7-11】 利用 KeyUp 事件中的 Shift 参数判定输入的英文字符大小写形式并加以显示。

KeyUp 事件中的 KeyCode 参数是按键的物理键码, 无论大写形式还是小写形式的字符其对应的 KeyCode 是相同的。例如, 输入字符 A 和字符 a, KeyCode 的值都是 65。为了区分输入的是大写形式还是小写形式, 应利用 Shift 参数检测 Shift 键是否被按下, 若被按下则为大写形式, 否则为小写形式。大写形式的字符其 ASCII 码值与物理键码相同, 可直接利用函数 Chr() 将其转换为相应的字符输出; 小写形式的字符其 ASCII 码值与物理键码相差 32, 故将物理键码加上 32 后进行转化输出。

窗体的 KeyUp 事件过程如下:

```
Private Sub Form_ KeyUp(KeyCode As Integer, Shift As Integer)
    ShiftDown = (Shift And vbShiftMask) > 0
    If KeyCode > = 65 And KeyCode < = 90 Then '若为 26 个字母键
        If ShiftDown Then
            Print Chr(KeyCode) '大写字符直接转换
        Else
            Print Chr(KeyCode + 32) '小写字符需要加 32 后转换
        End If
    End If
End Sub
```

在 VB 中已经把键盘上的功能键定义为常量, 即 vbKeyFX, 这里的 X 可以是 1 ~ 12 的值。例如, vbKeyF5 表示功能键 F5。这些功能键可以在程序中使用。以下语句表示按下功能键 F5 结束程序: If KeyCode = vbKeyF5 Then End。

以上, 介绍了键盘事件, 如果一个窗体的 KeyPreview 属性被设置为 True, 则该窗体将先于处在该窗体上的控件接收到键盘事件。因此, 可利用 KeyPreview 属性来创建全局键盘处理例程。

【例 7-12】 如图 7-20 所示, 利用 KeyPreview 属性创建全局键盘处理例程, 使输入的用户名和密码自动转换为大写字母。

程序代码如下:

```
Private Sub Form_ KeyPress(KeyAscii As Integer)
    If KeyAscii > = 97 And KeyAscii < = 122 Then
        KeyAscii = KeyAscii - 32
    End Sub
Private Sub Form_Load()
    Text1. Text = ""
    Text2. Text = ""
    KeyPreview = True
End Sub
```

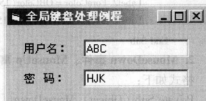


图 7-20 全局键盘处理例程

7.3.2 鼠标事件

鼠标也是一种人机交互工具。鼠标事件是由移动鼠标、按下或释放鼠标按键等操作产生的, 与鼠标有关的基本事件是 Click (单击)、DbClick (双击)、MouseDown (按下鼠标按键)、MouseUp (释放鼠标按键) 和 MouseMove (移动鼠标)。

1. Click 事件和 DbClick 事件

格式如下:

```
Private Sub <Object>_Click([Index As Integer])
```

```
End Sub
```

```
Private Sub <Object>_DblClick([Index As Integer])
```

```
...
```

```
End Sub
```

说明:

1) 释放任意鼠标按键,若鼠标光标所指对象是原按下鼠标按键时所指对象,则引发该对象的 Click 事件;双击任意鼠标按键会引发鼠标光标所指对象的 DbClick 事件。VB 提供了响应此两事件的事件过程的框架,其格式如上所示。

2) <Object>: Form、MDIForm 或其他可以响应鼠标事件的控件的名字。

3) Index: 整数值。当对象为 Form、MDIForm 或单个控件时不含此参数。当对象为控件数组的成员时,此参数是该成员控件在数组中的下标。

【例 7-13】如图 7-21 所示,在窗体中布置两个标签控件。令 Label1 显示“鼠标单击”、Label2 显示“鼠标双击”。分别响应 Label1 的 Click 事件和 Label2 的 DblClick 事件,从中随机改变两个标签控件的前景色和背景色。

程序代码如下:

```
Private Sub Label1_Click()
```

```
Label1.ForeColor = QBColor(15 * Rnd)
```

```
Label1.BackColor = QBColor(15 * Rnd)
```

```
End Sub
```

```
Private Sub Label2_DblClick()
```

```
Label2.ForeColor = QBColor(15 * Rnd)
```

```
Label2.BackColor = QBColor(15 * Rnd)
```

```
End Sub
```

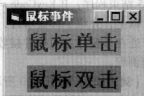


图 7-21 Click、DblClick 事件演示

2. MouseDown 事件、MouseUp 事件和 MouseMove 事件

格式如下:

```
Private Sub <Object>_MouseDown([Index As Integer],[Button As Integer],
```

```
Shift As Integer,X As Single,Y As Single)
```

```
...
```

```
End Sub
```

```
Private Sub <Object>_MouseUp([Index As Integer],[Button As Integer],
```

```
Shift As Integer,X As Single,Y As Single)
```

```
...
```

```
End Sub
```

```
Private Sub <Object>_MouseMove([Index As Integer],[Button As Integer],
```

```
Shift As Integer,X As Single,Y As Single)
```

```
...
```

```
End Sub
```

说明:

1) 按下任意鼠标按键会引发鼠标光标所指对象的 MouseDown 事件;释放任意鼠标按键会引发鼠标光标所指对象(或拖曳开始时鼠标光标所指对象)的 MouseUp 事件;移动鼠标光

标会引发鼠标光标所指对象(或拖曳开始时鼠标光标所指对象)的 MouseMove 事件。VB 提供了响应这些事件的事件过程的框架,其格式如上所示。

2) <Object>: Form、MDIForm 或其他可以响应鼠标事件的控件的名字。

3) Index: 整数值。当对象为 Form、MDIForm 或单个控件时不含此参数。当对象为控件数组的成员时,此参数是该成员控件在数组中的下标。

4) Button: 该参数为一个 3 位的二进制整数,表示鼠标的哪一个键被按下或放开,鼠标按键与此 3 位数的对应关系如表 7-15 所示。应注意,MouseDown 和 MouseUp 所使用的 Button 参数与 MouseMove 所使用的 Button 参数是不同的:对于 MouseDown 和 MouseUp 来说,Button 参数要精确地指出每个事件的一个按钮,即只能表示鼠标按键的单选情况,其 Button 参数的值只能为表 7-15 中 3 个符号常数中的一个;而对于 MouseMove 来说,它指示的是所有按钮的当前状态,即可以表示鼠标按键的多选情况,其 Button 参数的值除了可取表 7-15 中任意一个符号常数之外,其值还可以是以上 3 个符号常数的任意之和,例如,Button 等于 3(vbLeft_Button + vbRight_Button)表示当前鼠标左键、右键均被按下,而 Button 等于 7 表示鼠标的 3 个键均被按下。

表 7-15 Button 参数的值

符号常数	二进制数	十进制数	含 义
vbLeft_Button	001	1	左键
vbRight_Button	010	2	右键
vbMiddle_Button	100	4	中键

5) Shift: 一个 3 位二进制整数,表示鼠标事件发生时,键盘上的 Shift、Ctrl 和 Alt 键是否被按下。其对应关系如表 7-14 所示。如前所述,Shift 值可以表示多选的情况。

6) 与表 7-14 中符号常数作用类似,表 7-15 中符号常数也可用作位屏蔽,用于测试哪一个鼠标按键被按下或放开。例如,以下代码用于测试哪一个鼠标按键和键盘上的 Ctrl 键被一起按下。

```

LeftDown = (Button And vbLeftButton) > 0
RightDown = (Button And vbRightButton) > 0
MiddleDown = (Button And vbMiddleButton) > 0
CtrlDown = (Shift And vbCtrlMask) > 0
If (CtrlDown And LeftDown) Then Print "Ctrl 键和鼠标左键被一起按下"
If (CtrlDown And RightDown) Then Print "Ctrl 键和鼠标右键被一起按下"
If (CtrlDown And MiddleDown) Then Print "Ctrl 键和鼠标中键被一起按下"

```

7) X、Y: 这两个值对应于当前鼠标的位置,采用 ScaleMode 属性指定的位置。

【例 7-14】 鼠标事件演示的设计界面如图 7-22a 所示,在窗体中布置 3 个标签控件 Label1 ~ Label3,一个图像框控件 Image1。各控件的属性设置如表 7-16 所示。运行程序,在图像框上移动鼠标,3 个标签控件实时显示出鼠标光标所指图片像素颜色的三原色的值,如图 7-22b 所示。按住键盘上的 Ctrl 键,同时在图像框上单击左键,在弹出的 InputBox 中输入其

他图片文件的路径名, 可在图像框中打开其他图片; 按住键盘上的 Ctrl 键, 同时在图像框上单击右键, 将结束程序。

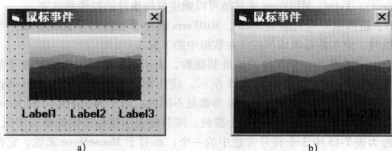


图 7-22 鼠标事件演示

a) 设计界面 b) 运行界面

表 7-16 例 7-14 各控件的属性设置

控件名称	属性名	属性值	控件名称	属性名	属性值
Form1	Caption	鼠标事件	Label3	AutoSize	True
	Height	2115		FontBold	True
	Width	2610		BackStyle	0—Transparent
	BorderStyle	1—Fixed Single	Image1	Stretch	True
Label1	AutoSize	True		Picture	C: \Documents and Settings \ AllUsers \ Documents \ My Pictures \ 示例图片 \ Blue hills.jpg
	FontBold	True			
	BackStyle	0—Transparent			
Label2	AutoSize	True			
	FontBold	True			
	BackStyle	0—Transparent			

程序代码如下:

```
Private Sub Form_Load()
```

将图像框 Image1 充满窗体的绘图区域

```
Image1.Left = Form1.ScaleLeft
```

```
Image1.Top = Form1.ScaleTop
```

```
Image1.Width = Form1.ScaleWidth
```

```
Image1.Height = Form1.ScaleHeight
```

```
End Sub
```

```
Private Sub Image1_MouseMove(Button As Integer, Shift As Integer, X As Single, _Y As Single)
```

```
c = Point(X, Y)
```

'获取相应点的颜色值

```
r = c Mod 256
```

'计算三原色中红色的值

```

g = c \ 256 Mod 256          '计算三原色中绿色的值
b = c \ 256 \ 256 Mod 256    '计算三原色中蓝色的值
Label1.Caption = "R = " & r   '显示结果
Label2.Caption = "G = " & g
Label3.Caption = "B = " & b

End Sub

Private Sub Image1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    LeftDown = (Button And vbLeftButton) > 0
    RightDown = (Button And vbRightButton) > 0
    CtrlDown = (Shift And vbCtrlMask) > 0
    If (CtrlDown And LeftDown) Then
        '如果按下 Ctrl 键的同时按下鼠标左键,则可以打开新的图片文件
        Image1.Picture = LoadPicture(InputBox("请输入图片的文件名","打开图片"))
    ElseIf (CtrlDown And RightDown) Then '如果按下 Ctrl 键的同时按下鼠标右键,则退出程序
    End
End If
End Sub

```

7.4 应用举例

本章在前面几节介绍了绘图操作及键盘和鼠标操作的知识,下面给出更多的应用例子,以巩固所学的知识。

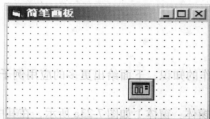
【例 7-15】 设计一个简笔画板。该程序包含两个窗体:窗体 1 如图 7-23a 所示,在其中添加 1 个通用对话框控件 Common Dialog1;窗体 2 如图 7-23b 所示,在其中添加 6 个标签控件 Label1 ~ Label6, 5 个直线控件 Line1 ~ Line5, 1 个文本框控件 Text1, 2 个按钮控件 Command1 和 Command2。各控件的属性设置见表 7-17。程序运行时,用户在窗体 1 的绘图区域按下鼠标左键并移动鼠标,程序将调用 PSet 方法,按鼠标移动轨迹画出行线条;按下鼠标中键,可以在弹出的窗体 2 中设置“画笔”宽度;按下鼠标右键,可以在弹出的标准颜色对话框中(利用通用对话框控件的 ShowColor 方法打开)设置“画笔”的颜色。程序运行界面如图 7-23c、d 所示。

表 7-17 例 7-15 各控件的属性设置

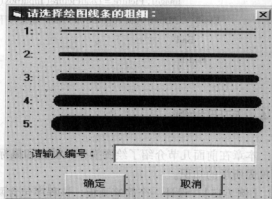
控件名称	属性名	属性值	控件名称	属性名	属性值
Form1	AutoRedraw	True	Form2.Label1	AutoSize	True
	BackColor	vbWhite		Caption	1:
	Caption	简笔画板		FontBold	True
	DrawWidth	2	Form2.Label2	AutoSize	True
Form1.CommonDialog1	CancelError	False		Caption	2:
Form2	BorderStyle	1—Fixed Single		FontBold	True

(续)

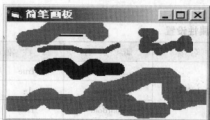
控件名称	属性名	属性值	控件名称	属性名	属性值
Form2. Label3	AutoSize	True	Form2. Line1	BorderWidth	2
	Caption	3:	Form2. Line2	BorderWidth	5
	FontBold	True	Form2. Line3	BorderWidth	10
Form2. Label4	AutoSize	True	Form2. Line4	BorderWidth	15
	Caption	4:	Form2. Line5	BorderWidth	20
	FontBold	True	Form2. Text1	Text	空
Form2. Label5	AutoSize	True	Form2. Command1	Caption	确定
	Caption	5:	Form2. Command2	Caption	取消
	FontBold	True			
Form2. Label6	AutoSize	True			
	Caption	请输入编号:			



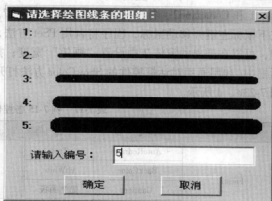
a)



b)



c)



d)

图 7-23 简笔画板程序界面

a) 设计界面1 b) 设计界面2 c) 运行界面1 d) 运行界面2

程序代码如下：

1) 窗体 1 程序代码：

```
'变量 IsPaint 为 True, 表示程序处于绘图状态; 为 False, 表示处于停止绘图状态
Dim IsPaint As Boolean

'变量 pColor 用于定义“画笔”的颜色值
Dim pColor As Long

Private Sub Form_ MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    LeftDown = (Button And vbLeftButton) > 0
    MiddleDown = (Button And vbMiddleButton) > 0
    RightDown = (Button And vbRightButton) > 0
    If LeftDown Then
        '鼠标左键被按下, 程序进入绘图状态
        IsPaint = True
    ElseIf MiddleDown Then
        '鼠标中键被按下, 按模式对话框方式弹出窗体 2, 让用户设置“画笔”宽度
        Form2.Show 1, Form1
    ElseIf RightDown Then
        '鼠标右键被按下, 利用通用对话框控件的 ShowColor 方法
        '弹出标准颜色对话框, 让用户设置“画笔”颜色
        CommonDialog1.ShowColor
        '用标准颜色对话框的返回值设置“画笔”颜色
        pColor = CommonDialog1.Color
    End If
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    '鼠标移动时, 如果处于绘图状态(鼠标左键被按下), 则用 PSet 方法, 模拟绘制线条
    '线条的颜色由变量 pColor 决定, 线条的宽度由 Form1 的 DrawWidth 属性值决定
    If IsPaint = True Then PSet(X, Y), pColor
End Sub

Private Sub Form_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    '鼠标按键被释放, 则退出程序的绘图状态
    IsPaint = False
End Sub
```

2) 窗体 2 程序代码：

```
Private Sub Command1_Click()
    '根据用户的选择, 为“画笔”设置宽度
    '“画笔”宽度由 Form1 的 DrawWidth 属性值决定, 而该属性值则由
    '用户选择的直线控件的 BorderWidth 属性值决定
    Select Case Val(Text1.Text)
        Case 1
            Form1.DrawWidth = Line1.BorderWidth
        Case 2
```

```

Form1.DrawWidth = Line2.BorderWidth
Case 3
    Form1.DrawWidth = Line3.BorderWidth
Case 4
    Form1.DrawWidth = Line4.BorderWidth
Case 5
    Form1.DrawWidth = Line5.BorderWidth
End Select
Me.Hide '退出窗体 2
End Sub
Private Sub Command2_Click()
    Me.Hide
End Sub

```

【例 7-16】 设计一个简易 CAD 程序，可以实现画点、直线、矩形和圆，并且用户可以调整“画笔”的宽度和颜色。设计界面如图 7-24a 所示，在窗体 Form1 中添加 1 个通用对话框 Common Dialog1，1 个图片框控件 Picture1，1 个文本框控件 Text1，1 个选项按钮组 Option1(0) ~ Option1(8)，1 个命令按钮组 Command1(1) ~ Command1(4)，2 个命令按钮 Command2、Command3，2 个标签控件 Label1、Label2。各控件的属性设置见表 7-18。

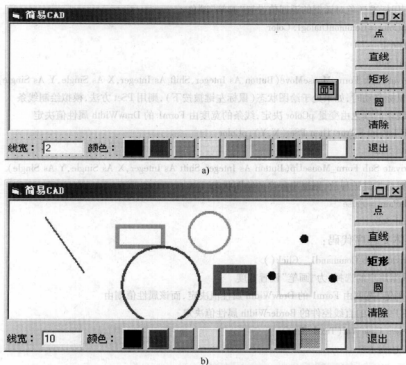


图 7-24 简易 CAD 程序界面
a) 设计界面 b) 运行界面

程序运行界面如图 7-24b 所示。程序运行时, 用户可以选择单击“点”、“直线”、“矩形”和“圆”4 个命令按钮之一(被选中的绘图按钮表面文字将被加粗), 实现相应图形的绘制。用户在绘制图形时, 对于绘制点, 只需在绘图区域中单击一次鼠标即可完成; 对于绘制直线、矩形和圆, 需要在绘图区域中两次单击鼠标(以确定直线的起点和终点、矩形对角的起止点、圆的圆心和半径)。

表 7-18 例 7-16 各控件的属性设置

控件名称	属性名	属性值	控件名称	属性名	属性值
Form1	Caption	简易 CAD	Option1 (6)	BackColor	vbBlue
CommonDialog1	CancelError	False		Caption	空
Picture1	BackColor	vbWhite	Option1 (7)	BackColor	&HFF00FF
Text1	Text	2		Caption	空
Option1 (0)	BackColor	vbBlack	Option1 (8)	BackColor	vbWhite
	Caption	空		Caption	空
Option1 (1)	BackColor	vbRed		FontBold	True
	Caption	空	Command1 (1)	Caption	点
Option1 (2)	BackColor	&H80FF&	Command1 (2)	Caption	直线
	Caption	空	Command1 (3)	Caption	矩形
Option1 (3)	BackColor	vbYellow	Command1 (4)	Caption	圆
	Caption	空	Command2	Caption	清除
Option1 (4)	BackColor	vbGreen	Command3	Caption	退出
	Caption	空		Label1	Caption
Option1 (5)	BackColor	vbCyan	Label2	Caption	颜色:
	Caption	空			

同时, 用户可以在线宽设置文本框中输入宽度值, 实现“画笔”宽度的设置; 在颜色设置选项按钮数组中单击欲使用的颜色, 实现“画笔”颜色的设置。设置画笔颜色时, 自左至右, 前 8 个颜色设置按钮是程序提供的标准颜色, 分别对应黑、红、橙、黄、绿、青、蓝、紫 8 种颜色。最后一个颜色设置按钮是自定义颜色设置按钮(默认为白色), 单击该按钮, 将弹出标准颜色对话框(利用通用对话框控件的 ShowColor 方法打开), 从中可选择自己需要的颜色。选中颜色后, 单击“确定”按钮退出颜色对话框, 该自定义颜色设置按钮的表面颜色将变成用户刚刚自定义的颜色。

单击“清除”按钮, 可以清空绘图区域; 单击“退出”按钮, 可以退出程序。

本题的难点在于绘制直线、矩形和圆时, 如何在第一次单击之后动态地让直线跟着鼠标移动, 直到第二次单击为止。需要在鼠标的 MouseDown 事件中捕捉到鼠标的当前位置, 在确定已经第一次单击之后, 在 MouseMove 事件中捕捉到鼠标当前的位置并画线、矩形或圆。由于鼠标不停地在移动, 而看到的直线、矩形或圆只应该有一条, 所以这里要不不停的重画, 并且清除刚才在 MouseMove 中留下的“痕迹”。此外, 对于绘图区域中那些已经画好的图形, 还需要进行还原, 因为清除只能用 Cls 方法, 这会把所有已经画好的图形全部清空。所

以对于已经画好的点、直线、矩形、圆，必须将它们存储起来，以便在需要的时候把它们重新绘制到图片框上。

为解决这个问题，可自定义 4 种数据类型，分别用来存放已经在图片框上所画的点、直线、矩形和圆的信息。同时为这 4 种类型定义 4 个自定义动态数组，每画一个图形元素就在相应类型的图形数组中添加一个新元素。当需要重画时，遍历 4 个数组，来实现重画。

程序代码如下：

Option Base 1 '改变默认数组下标开始为 1

Private Type Dots '自定义类型用来保存一个点的相关信息

x1 As Single '包括点的坐标、线宽、颜色

y1 As Single

Width As Integer

Color As Long

End Type

Private Type Lines '自定义类型用来保存一条直线的相关信息

x1 As Single '包括起止点、线宽、颜色

y1 As Single

x2 As Single

y2 As Single

Width As Integer

Color As Long

End Type

Private Type Rectangles '自定义类型用来保存一个矩形的信息

x1 As Single '包括对角线起止点、线宽、颜色

y1 As Single

x2 As Single

y2 As Single

Width As Integer

Color As Long

End Type

Private Type Circles '自定义类型用来保存一个圆的信息

x1 As Single '包括圆心、半径、线宽、颜色

y1 As Single

R As Single

Width As Integer

Color As Long

End Type

Dim DotStore() As Dots '用来存放已经画好的点的数组

Dim LineStore() As Lines '用来存放已经画好的直线的数组

Dim RectangleStore() As Rectangles '用来存放已经画好的矩形的数组

Dim CircleStore() As Circles '用来存放已经画好的圆的数组

Dim DotMaxIndex As Long '存放点数组的当前最大下标

Dim LineMaxIndex As Long '存放直线数组的当前最大下标

```

Dim RectangleMaxIndex As Long '存放矩形数组的当前最大下标
Dim CircleMaxIndex As Long '存放圆数组的当前最大下标
Dim x1 As Single, y1 As Single '保存直线的起点或矩形对角线的起点或圆的圆心位置
'变量 DrawGraph 表示绘图类型
'1 表示画点, 2 表示画直线, 3 表示画矩形, 4 表示画圆
Dim DrawGraph As Integer
'变量 IsPaint 为 True, 表示程序处于绘图状态; 为 False, 表示处于停止绘图状态
Dim IsPaint As Boolean
'变量 pColor 用于定义“画笔”的颜色值
Dim pColor As Long
Private Sub RedrawGraph() '自定义重画过程
    Dim i As Integer
    If DotMaxIndex Then '如果存放点的数组不为空, 则把其中的点都画出来
        For i = 1 To DotMaxIndex
            Picture1.DrawWidth = DotStore(i).Width
            Picture1.PSet (DotStore(i).x1, DotStore(i).y1), DotStore(i).Color
        Next i
    End If
    If LineMaxIndex Then '如果存放直线数组不为空, 则把其中的直线都画出来
        For i = 1 To LineMaxIndex
            Picture1.DrawWidth = LineStore(i).Width
            Picture1.Line (LineStore(i).x1, LineStore(i).y1) -(LineStore(i).x2, LineStore(i).y2) _
                , LineStore(i).Color
        Next i
    End If
    If RectangleMaxIndex Then '如果存放矩形的数组不为空, 则把其中的矩形都画出来
        For i = 1 To RectangleMaxIndex
            Picture1.DrawWidth = RectangleStore(i).Width
            Picture1.Line (RectangleStore(i).x1, RectangleStore(i).y1) -(RectangleStore(i).x2, _
                RectangleStore(i).y2), RectangleStore(i).Color, B
        Next i
    End If
    If CircleMaxIndex Then '如果存放圆数组不为空, 则把其中的圆都画出来
        For i = 1 To CircleMaxIndex
            Picture1.DrawWidth = CircleStore(i).Width
            Picture1.Circle (CircleStore(i).x1, CircleStore(i).y1), CircleStore(i).R, _
                CircleStore(i).Color
        Next i
    End If
    Picture1.DrawWidth = Text1.Text '恢复线宽设置
End Sub
'自定义函数过程求两点间距离
Private Function Distance(x1 As Single, y1 As Single, x2 As Single, y2 As Single)

```

```

Distance = Sqr((x1 - x2)^2 + (y1 - y2)^2)
End Function

Private Sub Command1_Click(Index As Integer)
    DrawGraph = Index '设置绘图类型
    '将选中的绘图按钮的文字加粗
    For Each com In Command1
        com.FontBold = False
    Next
    Command1(Index).FontBold = True
End Sub

Private Sub Command2_Click()
    '清除图片框内容
    Picture1.Cls
    '清除 4 个图形数组
    Erase DotStore
    Erase LineStore
    Erase RectangleStore
    Erase CircleStore
    '将 4 个图形数组的当前最大下标清 0
    DotMaxIndex = 0
    LineMaxIndex = 0
    RectangleMaxIndex = 0
    CircleMaxIndex = 0
End Sub

Private Sub Command3_Click()
    End '退出程序
End Sub

Private Sub Option1_Click(Index As Integer)
    Select Case Index
        Case 0 '选择黑色
            pColor = vbBlack
        Case 1 '选择红色
            pColor = vbRed
        Case 2 '选择橙色
            pColor = &H80FF&
        Case 3 '选择黄色
            pColor = vbYellow
        Case 4 '选择绿色
            pColor = vbGreen
        Case 5 '选择青色
            pColor = vbCyan
        Case 6 '选择蓝色
            pColor = vbBlue
    End Select
End Sub

```

Case 7 '选择紫色

pColor = &HFF00FF

Case 8 '选择自定义颜色, 在 MouseUp 事件中处理

End Select

End Sub

```
Private Sub Option1_MouseUp(Index As Integer, Button As Integer, _Shift As Integer, X As Single,
                             Y As Single)
```

LeftDown = (Button And vbLeftButton) > 0

If Index = 8 And LeftDown Then '选择自定义颜色, 如下处理

CommonDialog1.ShowColor

Option1(8).BackColor = CommonDialog1.Color

pColor = Option1(8).BackColor

End If

End Sub

```
Private Sub Picture1_MouseDown(Button As Integer, _Shift As Integer, X As Single, Y As Single)
```

LeftDown = (Button And vbLeftButton) > 0

If LeftDown Then

Select Case DrawGraph

Case 1 '如果是画点, 则如下处理

Picture1.PSet(X, Y), pColor

'对点的操作直接加入数组

ReDim Preserve DotStore(DotMaxIndex + 1)

DotMaxIndex = DotMaxIndex + 1

DotStore(DotMaxIndex).x1 = X

DotStore(DotMaxIndex).y1 = Y

DotStore(DotMaxIndex).Width = Picture1.DrawWidth

DotStore(DotMaxIndex).Color = pColor

Case 2 '如果是画直线, 则如下处理

If IsPaint Then

'如果已经有了起点, 则此次单击为终止画线

'并把最新所画的直线加入数组

Picture1.Line(x1, y1)-(X, Y), pColor

ReDim Preserve LineStore(LineMaxIndex + 1)

LineMaxIndex = LineMaxIndex + 1

LineStore(LineMaxIndex).x1 = x1

LineStore(LineMaxIndex).y1 = y1

LineStore(LineMaxIndex).x2 = X

LineStore(LineMaxIndex).y2 = Y

LineStore(LineMaxIndex).Width = Picture1.DrawWidth

LineStore(LineMaxIndex).Color = pColor

Else '如果是第一次单击, 则用全局变量记录下当前的点, 暂不加入数组

x1 = X



```

y1 = Y
End If
IsPaint = Not IsPaint '设置绘图状态开关变量
Case 3 '如果是画矩形,则如下处理
If IsPaint Then
'如果已经有了对角线起点,则此次单击为终止画矩形
'并把最新所画的矩形加入数组中
Picture1.Line (x1, y1)-(X, Y), pColor, B
ReDim Preserve RectangleStore(RectangleMaxIndex + 1)
RectangleMaxIndex = RectangleMaxIndex + 1
RectangleStore(RectangleMaxIndex).x1 = x1
RectangleStore(RectangleMaxIndex).y1 = y1
RectangleStore(RectangleMaxIndex).x2 = X
RectangleStore(RectangleMaxIndex).y2 = Y
RectangleStore(RectangleMaxIndex).Width = Picture1.DrawWidth
RectangleStore(RectangleMaxIndex).Color = pColor
Else '如果是第一次单击,则用全局变量记录下当前的点,暂不加入数组
x1 = X
y1 = Y
End If
IsPaint = Not IsPaint '设置绘图状态开关变量
Case 4 '如果是画圆,则如下处理
If IsPaint Then
'如果已经有了圆心,则此次单击为终止画圆,并把最新所画的圆加入数组中
Picture1.Circle (x1, y1), Distance(x1, y1, X, Y), pColor
ReDim Preserve CircleStore(CircleMaxIndex + 1)
CircleMaxIndex = CircleMaxIndex + 1
CircleStore(CircleMaxIndex).x1 = x1
CircleStore(CircleMaxIndex).y1 = y1
CircleStore(CircleMaxIndex).R = Distance(x1, y1, X, Y)
CircleStore(CircleMaxIndex).Width = Picture1.DrawWidth
CircleStore(CircleMaxIndex).Color = pColor
Else '如果是第一次单击,则用全局变量记录下当前的点,暂不加入数组
x1 = X
y1 = Y
End If
IsPaint = Not IsPaint '设置绘图状态开关变量
End Select
End If
End Sub

Private Sub Picture1_MouseMove(Button As Integer, _Shift As Integer, X As Single, Y As Single)
If IsPaint = True Then '如果处于绘图状态,则如下处理

```

Select Case DrawGraph

Case 1 ' 如果是画点,则不需响应

Case 2 ' 如果是画直线,则如下处理

Picture1.Cls '先清屏

Picture1.Line (x1, y1)-(X, Y), pColor '画当前直线

RedrawGraph '重画屏幕上原来的内容

Case 3 '如果是画矩形,则如下处理

Picture1.Cls '先清屏

Picture1.Line (x1, y1)-(X, Y), pColor, B '画当前矩形

RedrawGraph '重画屏幕上原来的内容

Case 4 '如果是画圆,则如下处理

Picture1.Cls '先清屏

Picture1.Circle (x1, y1), Distance(x1, y1, X, Y), pColor '画当前圆

RedrawGraph '重画屏幕上原来的内容

End Select

End If

End Sub

Private Sub Text1_Change()

Picture1.DrawWidth = Text1.Text '线宽设置

End Sub

习 题

- 7-1 单击窗体时,以窗体中心点为起点,随机画 100 条直线,产生“放花”的效果,如图 7-25 所示。
7-2 在窗体上以随机的颜色从外向里画矩形,每隔 1s 画一个,如图 7-26 所示。



图 7-25 随机画线



图 7-26 以随机的颜色画矩形

- 7-3 用 Pset 方法在窗体上绘制一条如图 7-27 所示的星形曲线,曲线由以下参数方程决定:

$$x = \sin 2t * \cos t$$

$$y = \sin 2t * \sin t$$

其中 t 的取值范围为 $0 \leq t \leq 2\pi$ 。

- 7-4 如图 7-28 所示,在窗体中画一个米字形。当窗体的大小改变时,米字形也随着自动调整。
7-5 单击窗体时,用 Pset 方法在窗体上绘制一条 $[0^\circ, 360^\circ]$ 的正弦曲线,如图 7-29 所示。
7-6 单击窗体时,在窗体上随机画一些带颜色的点,实现满天星的效果,如图 7-30 所示。

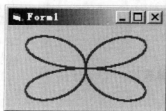


图 7-27 画星形曲线

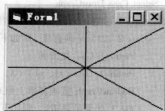


图 7-28 画米字形

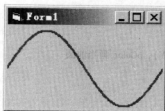


图 7-29 画正弦曲线



图 7-30 随机画带颜色的点

7-7 编写一个程序，当同时按下 Alt、Shift 和 A 键时，在窗体上画一个圆；当同时按下 Alt、Shift 和 B 键时，清空窗体。

7-8 编写一个程序，当按下键盘上的某个键时，在窗体上输出该键的 KeyCode 码。

7-9 编写一个程序，使窗体上的文本框中只能接收 0~9 的数字字符。如果输入了其他字符，则响铃 (Beep)，并且消除该字符。

7-10 编写一个程序，使用户按下功能键 F1 时，显示一个标题栏内容为“帮助窗口”的窗体。

7-11 设计一个最简单的画图程序，要求程序运行时，按下鼠标左键画圆，按下鼠标右键移动时画线。



第8章 文件操作

在前面的各章中，所用的输入和输出都是以显示器和键盘为对象，即以键盘将数据输入，并将计算机的处理结果从显示器输出。这种简单的输入和输出只能解决一些比较简单的问题。当需要计算机处理更加复杂的问题时，这种输入和输出方法就不能满足用户的要求，就必须用文件进行处理。

8.1 文件

文件(File)是程序设计中的一个重要概念。所谓“文件”，一般指存储在计算机外部介质上数据的集合。操作系统是以文件为单位对数据进行管理的，也就是说，想要找到存储在计算机外部介质上的数据，必须先按文件名找到所指定的文件，然后再从该文件中读取数据。要向外部介质上存储信息，也必须先建立一个文件(以文件名标识)，然后再向该文件输出数据。

VB 允许使用两种不同的方法来处理驱动器、文件夹和文件。

- 1) 使用诸如 Open、Write 语句等传统方法。
- 2) 使用新的工具 File System Object 对象模型。

8.1.1 文件结构

为了更有效地对数据进行存储和读取，文件中的数据必须以某种特定的格式去存储，这种特定的格式称为文件的结构。VB 文件由记录组成，记录由字段组成，字段由字符组成。

字符(Character)是构成文件的最基本单位，可以是字母、数字、特殊符号或单一字节。这里所说的“字符”一般为西文字符，一个西文字符用一个字节存放。如果是汉字字符，包括汉字和“全角”字符，则通常用两个字节存放。也就是说，一个汉字字符相当于两个西文字符。一般把用一个字节存放的西文字符称为“半角”字符，而把汉字和用两个字节存放的字符称为“全角”字符。注意，VB 支持双字节字符，当计算字符串长度时，一个西文字符和一个汉字都作为一个字符计算，但它们所占的内存空间是不一样的。例如，字符串“学说英语 ABC”的长度为 7，而所占的字节数为 11。

字段(Field)也称域，由若干个字符组成，用来表示一项数据。例如，姓名“赵连山”就是一个字段，它由 3 个汉字组成；而学号“20070102”也是一个字段，它由 8 个字符组成。

记录(Record)由一组相关的字段组成。例如，在学生档案中，每个人的学号、姓名、性别、年龄、班级等构成一个记录。VB 中的记录见表 8-1。在 VB 中，以记录为单位处理数据。

表 8-1 VB 中的记录

学 号	姓 名	性 别	年 龄	班 级
20070101	李金霞	女	18	2



在 VB 文件中, 一个文件含有一个以上的记录。例如, 在学生档案文件中有 1000 个人的信息, 每个人的信息是一个记录, 1000 个记录构成一个文件。

8.1.2 文件类型

按照不同的标准, 文件可分为不同的类型。

1. 按照数据性质分类

按照数据性质, 文件可分为程序文件和数据文件。

1) 程序文件(Program File): 这种文件存放的是可以由计算机执行的程序, 包括源文件和可执行文件。扩展名为“.exe”、“.frm”、“.vbp”、“.bas”、“.cls”等的文件都是程序文件。

2) 数据文件(Data File): 数据文件用来存放普通的数据。例如学生考试成绩、职工工资、商品库存等。这类数据必须通过程序来存取和管理。

2. 按照数据的存取方式和结构分类

按照数据的存取方式和结构, 文件可分为顺序文件和随机存取文件。

1) 顺序文件(Sequential File): 顺序文件的结构比较简单, 文件中的记录一个接一个地存放。在这种文件中, 当要查找某个数据时, 只能从文件头开始, 一个记录一个记录地顺序读取, 直至找到要查找的记录为止。

优点: 顺序文件的组织比较简单, 只要把数据记录一个接一个地写到文件中即可, 占用空间少, 容易使用。

缺点: 维护困难, 为了修改文件中的某个记录, 必须把整个文件读入内存, 修改完后重新写入磁盘。顺序文件不能灵活地存取和增减数据, 因而适用于有一定规律且不经常修改的数据。

2) 随机存取文件(Random Access File): 又称直接存取文件, 简称随机文件或直接文件。在随机文件中, 每个记录的长度是固定的, 记录中的每个字段的长度也是固定的。此外, 随机文件的每个记录都有一个记录号。在写入数据时, 只要指定记录号, 就可以把数据直接存入指定位置。而在读取数据时, 只要给出记录号, 就能直接读取该记录。在随机文件中, 可以同时进行读、写操作, 因而能快速地查找到每个记录, 不必为修改某个记录而对整个文件进行读、写操作。

优点: 数据的存取较为灵活、方便, 速度较快, 容易修改。

缺点: 占空间较大, 数据组织较复杂。

3. 按照数据的编码方式分类

按照数据的编码方式, 文件可以分为 ASCII 文件和二进制文件。

1) ASCII 文件: 又称文本文件, 它是以 ASCII 方式保存的文件。这种文件可以用字处理软件建立和修改(必须按纯文本文件保存)。

2) 二进制文件(Binary File): 它是用二进制方式保存的文件。二进制文件不能用普通的字处理软件编辑, 占空间较小。

8.1.3 文件的打开与关闭

在 VB 中, 对文件的操作按下述步骤进行:

1) 打开或建立文件。一个文件必须先打开或建立后才能使用。如果一个文件已经存在,则打开该文件;如果不存在,则建立该文件。该操作的结果是建立一个(内存中的)文件缓冲区并与特定文件关联。例如,执行语句 `Open "c:\VB\test" For Output As #1` 后,VB 应用程序即可将文件缓冲区 1(文件号#1)作为内部数据记录与外部文件 `c:\VB\test` 记录的数据交换区。

2) 读/写操作。用文件读/写语句读/写文件缓冲区,在打开或建立的文件上执行所要求的输入/输出操作。在文件处理中,把内存中的数据传输到相关的外部设备(例如磁盘)并作为文件存放的操作叫做写数据,而把数据文件中的数据传输到内存中的操作叫做读数据。一般来说,在主存与外设的数据传输中,由主存到外设叫做输出或写,而由外设到主存叫做输入或读。

3) 关闭文件。关闭文件操作将使(内存中)文件缓冲区的记录数据写入外部文件。

文件处理一般需要以上 3 个步骤。在 VB 中,数据文件的操作是通过有关的语句和函数来实现的。

1. 文件的打开(建立)

在 VB 中,可以用 `Open` 语句打开或建立一个文件。

格式如下:

`Open <文件名> [For <访问方式>] [Access <存取类型>] [<锁定>] As [#] <文件号> [Len = <记录长度>]`

功能:打开或建立一个文件,为文件的输入输出分配缓冲区,并确定缓冲区所使用的存取方式。

说明:

1) 格式中的“文件名”用于指定所要打开文件的文件名,包括目录、文件夹和驱动器。

2) 格式中的“访问方式”用于指定文件的输入/输出方式。

■ **Output**: 指定顺序输出方式,文件被打开后只能写数据。

■ **Input**: 指定顺序输入方式,文件被打开后只能读取数据。

■ **Append**: 指定顺序输出方式。与 **Output** 不同的是,当用 **Append** 方式打开文件时,如对文件执行写操作,则写入的数据附加到原来文件的后面。

■ **Random**: 指定随机存取方式,也是默认方式。

■ **Binary**: 指定二进制存取方式。在这种方式下,可以用 `Get` 和 `Put` 语句对文件中任何字节位置的信息进行读写。打开文件的类型与 **Random** 方式相同。

访问方式是可选的,如果省略,则为随机存取方式,即 **Random**。

3) 格式中的“存取类型”用于指定访问文件的类型。

■ **Read**: 打开只读文件。

■ **Write**: 打开只写文件。

■ **Read Write**: 打开读/写文件。这种类型只对随机文件、二进制文件及用 **Append** 方式打开的文件有效。

“存取类型”指出了在打开的文件中所进行的操作。如果要打开的文件已由其他过程打开,则不允许指定存取类型,否则打开失败,并产生错误信息。

4) 格式中的“文件号”是一个整型表达式,其值在 1~511 的范围内。文件一旦打开成功,系统将文件与文件号相关联,程序可直接使用文件号对文件进行操作。为了避免文件号的重叠使用,VB 提供了 FreeFile 函数用于为打开的文件分配系统中尚未被使用的文件号。文件号前的#号可以省略。

5) 格式中的“记录长度”是一个整型表达式。当选择该参数时,对于用随机访问方式打开的文件,该值是记录长度;对于顺序文件,该值是缓冲字符数。“记录长度”的值不能超过 32767B。对于二进制文件,将忽略 Len 子句。

在顺序文件中,“记录长度”不需要与各个记录的大小相对应,因为顺序文件各个记录的长度可以不相同。当打开顺序文件时,在把记录写入磁盘或从磁盘读出记录之前,“记录长度”指出要装入缓冲区的字符数,即确定缓冲区的大小。缓冲区越大,占用内存空间越多,文件的输入/输出操作越快。反之,缓冲区越小,占用的内存空间越小,文件的输入/输出操作越慢。默认时缓冲区的容量为 512B。

另外,为了满足不同的存取方式的需要,对同一个文件可以用几个不同的文件号打开,每个文件号有自己的一个缓冲区。对于不同的访问方式,可以使用不同的缓冲区。但是,当使用 Output 或 Append 方式时,必须先将文件关闭,才能重新打开文件。而当使用 Input、Random 或 Binary 方式时,不必关闭文件就可以用不同的文件号打开文件。

例如,下面打开的文件都是按顺序方式输入输出的:

```
Open "No1.dat" For Output As #1
```

建立或打开一个文件,使记录可以写到该文件中。如果文件“No1.dat”已存在,该语句打开已存在的数据文件,新写入的数据将覆盖原来的数据。

```
Open "No1.dat" For Append As #1
```

打开已存在的文件,新写入的记录附加到文件的后面,原来的数据仍在文件中。如果给定的文件名不存在,则 Append 方式可以建立一个新文件。

```
Open "No1.dat" For Input As #1
```

打开已存在的文件,以便从文件中读出记录。

例如,下面是按随机方式打开或建立文件:

```
Open "No1.dat" For Random As #1
```

按随机方式打开或建立文件,然后读出或写入定长记录。

```
Open "e:\temp\No1.dat" For Random As #1 Len = 256
```

用随机方式打开 C 盘上的 temp 文件夹内的 No1.dat 文件,记录长度为 256B。

2. 文件的关闭

文件的读写操作结束后,应将文件关闭,这可以通过 Close 语句来实现。

格式如下:

```
Close[[#]文件号][,[#][文件号]]……
```

功能:用来结束文件的输入输出操作。

说明:

1) Close 语句用来关闭文件,它是在打开文件之后进行的操作。格式中的“文件号”是 Open 语句中使用的文件号。关闭一个数据文件具有两方面的作用,第一,把文件缓冲区中的所有数据写到文件中;第二,释放与该文件相联系的文件号,以供其他 Open 语句

使用。

2) Close 语句中的“文件号”是可选的。如果指定了文件号,则把指定的文件关闭;如果不指定文件号,则把所有打开的文件全部关闭。

3) 除了用 Close 语句关闭文件外,在程序结束时将自动关闭所有打开的数据文件。

4) Close 语句使 VB 结束对文件的使用,它的操作十分简单,但绝不是可有可无的。这是因为,磁盘文件同内存之间的信息交换是通过缓冲区进行的。如果关闭的是顺序输出而打开的文件,则缓冲区中最后的内容将被写入文件中。当打开的文件或设备正在输出时,执行 Close 语句后,不会使输出信息的操作中断。如果不使用 Close 语句关闭文件,则可能使某些需要写入的数据不能从内存(缓冲区)送入文件中。

8.2 顺序文件

在顺序文件中,构成文件的记录不定长,记录与记录间有明确的分隔符。例如,文件 NameList:

1、“张峰”;	2、“赵茵茵”;	3、“Tom”;	…;	1000、“东方致远”
---------	----------	----------	----	-------------

就构成了一个顺序文件,每个记录间用“;”分隔。

由于记录不定长,无法直接定位各个记录的开始和结束。因此,要读写顺序文件记录,必须从文件头开始,一个记录一个记录地顺序进行。

顺序文件的读/写操作与标准输入/输出十分类似。其中,读操作是把文件中的数据读到内存,标准输入是从键盘上输入数据,而键盘设备也可以看做是一个文件;写操作是把内存中的数据输出到屏幕上,而屏幕设备也可以看做是一个文件。

8.2.1 顺序文件的写操作

顺序文件的写操作分为 3 步进行,即打开文件、写入文件和关闭文件。其中打开文件和关闭文件分别由 Open 和 Close 语句来实现,写入文件由 Print #语句或 Write #语句来完成。

1. Print #语句

格式如下:

Print # <文件号>,[[Spc(n)] [Tab(n)] [表达式表] [;|,]]

功能:将“表达式表”的数据值写入由“文件号”指定的顺序文件中。

说明:

1) 格式中的“文件号”的含义同前,数据被写入该文件号所代表的文件中。

2) Print #语句与 Print 方法的功能类似,Print 方法所“写”的对象是窗体、打印机或控件,而 Print #语句所“写”的对象是文件。格式中的 Spc 函数、Tab 函数、“表达式表”及尾部的分号、逗号等,其含义与 Print 方法中相同。

例如,Print #1, X, Y, Z 是将变量 X、Y、Z 的值写到文件号为 1 的文件中。

而 Print X, Y, Z 是将变量 X、Y、Z 的值“写”到窗体上。

3) 和 Print 方法一样,Print #语句中的各数据项之间可以用分号隔开,也可以用逗号隔

开, 分别对应紧凑格式和标准格式。数值数据由于前有符号位, 后有空格, 因此使用分号不会给以后读取文件造成麻烦。但是, 对于字符串数据, 特别是变长字符串中数据来说, 用分号分隔就有可能引起麻烦, 因为输出的字符串数据之间没有空格。

例如:

```
X$="One";Y$="Two";Z$="Three"
```

```
Print #1,X$;Y$;Z$
```

写到磁盘上的信息为 OneTwoThree。为了使输出的各字符中明显地分开, 可以人为地插入逗号, 即改为:

```
Print #1,X$",";Y$",";Z$
```

这样写入文件中的信息为 One, Two, Three。

但是, 如果字符串本身含有逗号、分号和有意义的前后空格及回车或换行, 则需用双引号 (ASCII 码 34) 作为分隔符, 把字符串放在双引号中写入磁盘。

例如:

```
X$="Tom;John;May"
```

```
Y$="123,456,789"
```

```
Print #1,Chr(34);X$;Chr(34);Chr(34);Y$;Chr(34)
```

写入文件的数据为 "Tom;John;May" "123,456,789"。

4) 格式中的“表达式”可以省略。在这种情况下, 将向文件中写入一个空行。

例如, Print #1。

5) 实际上, Print #语句的任务只是将数据送到缓冲区, 数据由缓冲区写到磁盘文件的操作是由文件系统来完成的。对于用户来说, 可以理解为由 Print #语句直接将数据写入磁盘文件。但是, 执行 Print #语句后, 并不是立即把缓冲区中的内容写入磁盘, 只有在满足下列条件之一时才写盘:

- 缓冲区已满;
- 缓冲区未满, 但执行下一个 Print #语句;
- 关闭文件 (Close)。

2. Write #语句

格式如下:

Write #文件号, 表达式表

功能: 将表达式表的数据值写入顺序文件, 各数据值间自动写入逗号分隔符。

说明:

1) 当使用 Write #语句时, 文件必须以 Output 或 Append 方式打开。“文件号”和“表达式表”的含义同前, “表达式表”中的各项以逗号分开。

2) Write #语句与 Print #语句的功能基本相同。二者区别有以下两点:

- 当用 Write #语句向文件写数据时, 数据在磁盘上以紧凑格式存放, 能自动地在数据项之间插入逗号, 并给字符串加上双引号。一旦最后一项被写入, 就插入新的一行。

Write #语句自动写入的逗号分隔符和回车换行符将有助于 Input #语句读取数据。

- 用 Write #语句写入的正数的前面没有空格。

例如:

```
Open "C:\Tmp222\No1. dat" For Output As #1
```

```
Open "C:\Tmp222\No2. dat" For Output As #2
```

```
X$ = "One"
```

```
Y$ = "Two"
```

```
Z$ = "Three"
```

```
Print #1, X$, Y$, Z$
```

```
Write #2, X$, Y$, Z$
```

```
Close #1, #2
```

写入 No1. dat 文件的数据为 One Two Three。

写入 No2. dat 文件的数据为 "One", "Two", "Three"。

【例 8-1】 编写程序，用 Print #语句向文件中写入数据。

```
Dim sNum, sName, sScore As String
```

```
Private Sub Form_Click()
```

```
Open "d:\VBtemp\Student. dat" For Output As #1
```

```
sNum = InputBox("请输入学号:", "数据输入")
```

```
sName = InputBox("请输入姓名:", "数据输入")
```

```
sScore = InputBox("请输入成绩:", "数据输入")
```

```
Print #1, sNum, sName, sScore
```

```
Close #1
```

```
End Sub
```

上述过程首先在 D 盘的 VBtemp 文件夹内打开一个名为 Student. dat 的输出 (Output) 文件，文件号为 1。然后在 3 个输入对话框中分别输入学号、姓名、成绩的内容，程序用 Print #语句把输入的数据写入文件 Student. dat 中。最后用 Close 语句关闭文件。

【例 8-2】 在 D 盘的 VBtemp 文件夹内建立一个 School. dat 文件，存放学校名称和该学校的电话号码。

```
Dim School, Tel As String
```

```
Private Sub Form_Click()
```

```
Open "d:\VBtemp\School. dat" For Output As #1
```

```
School = InputBox("请输入学校名称:")
```

```
While UCase(School) <> "END"
```

```
Tel = InputBox("请输入学校电话:")
```

```
Write #1, School, Tel
```

```
School = InputBox("请输入学校名称:")
```

```
Wend
```

```
Close #1
```

```
End Sub
```

上述程序反复地从键盘上输入学校名称和电话号码，并写到磁盘文件 School. dat 中，直到输入 END 为止。读者可以把用该程序建立的文件与前一个例子建立的文件进行比较，看它们有什么区别 (用“记事本”查看)。



如果需要向电话号码文件中续加新的电话号码,则需把操作方式由 Output 改为 Append,即把 Open 语句改为

```
Open "d:\VBtemp\School.dat" For Append As #1
```

实际上,由于 Append 方式具有建立文件的功能,因此最好在开始建立文件时就使用 Append 方式。

由 Open 语句建立的顺序文件是 ASCII 文件,可以用字处理软件来查看或修改。顺序文件由记录组成,每个记录是一个单一的文本行,它以回车换行序列结束。每个记录又被分成若干个字段,这些字段是记录中按同一顺序反复出现的数据块。在顺序文件中,每个记录可以具有不同的长度,不同记录中的字段的长度也可以不一样。

当把一个字段存入变量时,存储字段的变量的类型决定了该字段的开头和结尾。当把字段存入字符串变量时,下列符号表示该字符串的结尾:

- 双引号("): 当字符串以双引号开头时;
- 逗号(,): 当字符串不以双引号开头时;
- 回车—换行: 当字段位于记录的结束处时。

如果把字段写入一个数值变量,则下列符号表示该字段的结尾:

- 逗号;
- 一个或多个空格;
- 回车—换行。

【例 8-3】 从键盘输入 5 个学生的数据,然后将它们存放到磁盘文件 StudentInfo.dat 中。

学生的数据包含学号、姓名、年龄、籍贯,用一个记录类型来定义。按下述步骤编写程序:

1) 执行“工程”菜单中的“添加模块”命令,建立标准模块,定义如下记录类型:

```
Type Student
```

```
Num As String * 5
```

```
Name As String * 8
```

```
Age As Integer
```

```
Address As String * 10
```

```
End Type
```

将该模块以文件名 EXAM8_3.bas 存盘。

2) 在窗体层输入如下代码:

```
Option Base 1
```

3) 编写如下的窗体事件过程:

```
Private Sub Form_Click()
```

```
Static stu() As Student
```

```
Open "d:\VBtemp\StudentInfo.dat" For Output As #1
```

```
n = InputBox("请输入学生人数:")
```

```
ReDim stu(n) As Student
```

```
For i = 1 To n
```

```
stu(i).Num = InputBox("请输入学生学号:")
```

```

stu(i). Name = InputBox("请输入学生姓名:")
stu(i). Age = InputBox("请输入学生年龄:")
stu(i). Address = InputBox("请输入学生籍贯:")
Write #1, stu(i). Num, stu(i). Name, stu(i). Age, stu(i). Address
Next i
Close #1
End
End Sub

```

该过程首先定义一个记录数组(大小未定), 打开一个输出文件 StudentInfo. dat (在 D 盘 VBtemp 文件夹内)。接着询问要输入的学生人数, 输入后重新定义数组。然后用 For 循环从键盘上输入每个学生的学号、姓名、年龄和籍贯, 并用 Write #语句写入磁盘文件中。最后关闭文件, 退出程序。

程序运行后, 在输入对话框中, 输入学生人数, 输入 5, 并单击“确定”按钮后, 即开始输入每个学生的数据。5 个学生的数据输入完毕后, 程序结束。此时屏幕上并没有信息输出, 只是把从键盘上输入的数据写到磁盘文件中。可以在记事本中查看该文件的内容, 如图 8-1 所示。

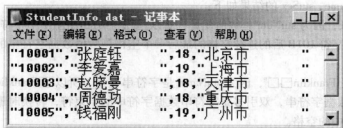


图 8-1 StudentInfo. dat 文件的内容

可以看出, 由于是用 Write #语句执行些操作, 文件中各项数据项之间用逗号隔开, 字符串数据放在双引号中。

8.2.2 顺序文件的读操作

顺序文件的读操作分为 3 步进行, 即打开文件、读顺序文件和关闭文件。其中打开文件和关闭文件分别由 Open 和 Close 语句来实现, 读顺序文件的操作由 Input #语句或 Line Input #语句来实现。

1. Input #语句

格式如下:

Input # <文件号>, <变量表>

功能: 从一个顺序文件中读出数据项, 并把这些数据项赋给程序变量。

例如:

```
Input #1, X, Y, Z
```

从文件中读出 3 个数据项, 分别把它们赋给 X、Y、Z 这 3 个变量。

说明:

1) 格式中的“文件号”的含义同前,数据项由该文件号所代表的文件中读出。

2) “变量表”由一个或多个变量组成,多个变量之间用逗号来分隔。这些变量既可以是数值变量,也可以是字符串变量或数组元素,从顺序文件中读出的数据赋给这些变量。文件中数据项的类型应与 Input #语句中变量的类型匹配。变量表中变量的个数一般等于文件的记录字段的个数。通常一次可以读出一条记录,以一条记录的结束符作为标志。

3) 在用 Input #语句把读出的数据项赋给变量时,将忽略前导空格、回车或换行符,把遇到的第一个非空格、非回车和换行符作为数据项的开始。对于数值变量所需的数据字符串以逗号、回车换行符或非空字符后的连续空格的最后一个空格结束;对于字符串变量所需的数据字符串以逗号、回车换行符或双引号结束(若数据字符串的第一个非空字符为双引号)。如果需要把开头带有空格的字符串赋给变量,则必须把字符串放在双引号中。

4) Input#语句与 InputBox 函数类似,但 InputBox 要求从键盘上输入数据,而 Input #语句要求从文件中输入数据,而且执行 Input #语句时不显示对话框。

5) Input #语句也可用于随机文件的读取。

例如,设 stuNum 为整型变量,stuName 和 stuSex 为字符串变量,#1 文件的前 35 个字符为“□□□101B□□"Franklin□□"Male□□18□□□<回车><换行>”,则执行 Input #1, stuNum, stuName, stuSex 的结果如下:

stuNum = 101, 因为 stuNum 是数值变量,所以□□□101B□□(9 个字符)构成第 1 个数据字符串,非空字符 101B 后的连续空格的最后一个空格是数值数据字符串的结束符,转换时删除前后空格。

stuName = "Franklin□□", 因为 stuName 是字符串变量,所以"Franklin□□"(13 个字符)构成第 2 个数据字符串,双引号是字符串数据字符串的结束符,转换时删除前后双引号但不删除双引号中的空格。

stuSex = "Male□□18", 因为 stuSex 是字符串变量,所以 Male□□18□□□<回车><换行>(13 个字符)构成第 3 个数据字符串,<回车><换行>是字符串数据字符串的结束符,空格不是字符串数据字符串的结束符,转换时删除<回车><换行>和前后空格。

【例 8-4】把例 8-3 建立的学生数据文件 StudentInfo. dat 读到内存,并在屏幕(窗体)上显示出来。

该程序的标准模块仍使用例 8-3 程序中的“EXAM8_3. bas”,窗体层代码也与例 8-3 相同。窗体事件过程如下:

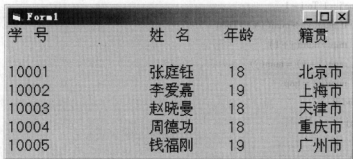
```
Private Sub Form_Click()  
    Static stu() As Student  
    Open "d:\VBtemp\StudentInfo. dat" For Input As #1  
    n = InputBox("请输入学生人数:")  
    ReDim stu(n) As Student  
    FontSize = 12  
    Print "学 号"; Tab(20); "姓 名"; Tab(30); "年龄"; Tab(40); "籍贯"  
    Print  
    For i = 1 To n
```

```

Input #1, stu(i). Num, stu(i). Name, stu(i). Age, stu(i). Address
Print stu(i). Num; Tab(20); stu(i). Name; Tab(30); stu(i). Age; Tab(40); stu(i). Address
Next i
Close #1
End Sub

```

该过程首先以输入方式打开文件 StudentInfo. dat，它是例 8-3 程序建立的学生数据文件，数组定义方式与前面的程序相同。在 For 循环中，用 Input #语句读入 5 个学生的数据，并在窗体上显示出来。在程序运行后，单击窗体，在输入对话框中输入 5，然后单击“确定”按钮，执行结果如图 8-2 所示。



学号	姓名	年龄	籍贯
10001	张庭钰	18	北京市
10002	李爱嘉	19	上海市
10003	赵晓曼	18	天津市
10004	周德功	18	重庆市
10005	钱福刚	19	广州市

图 8-2 数据文件的读出与显示执行结果

当需要输入大批量的数据时，如果用 InputBox 函数一个一个地输入，效率太低。现在，可以通过 Input #语句从文件中读取数据来解决，如例 8-5 所述。

【例 8-5】编写程序，对数值数据排序。

排序算法有很多种，第 5 章曾介绍过冒泡法。下面用冒泡法对数值数据进行排序。需要排序的数据放在一个数据文件中，名为 Data. dat，内容如下：

50

```

411  66  159  -421 -397  549  -972  521  628  418
-910 -172  725  580  -253  923  742  -888  899  -272
49   534  -893  184  -63  -404  245  295  -473  -442
659  649  178  972  821  -547  390  960  -513  67
-788  998  352  -969  150  -800  -794  597  -432  -909

```

文件中共有 51 个数值数据，第一个数据 50 表示数据个数，实际参加排序的有 50 个数据。各数据之间用空格或回车符分开。可以用“记事本”程序建立 Data. dat 文件。

窗体层编写如下代码：

```
Option Base 1
```

编写如下事件过程：

```
Private Sub Form_Click()
```



```
Static num() As Integer
```

```
Open "d:\VBtemp\Data.dat" For Input As #1
```

```
Input #1, n
```

```
ReDim num(n) As Integer
```

```
FontSize = 12
```

```
For i = 1 To n
```

```
Input #1, num(i)
```

```
Next i
```

```
For i = n To 2 Step -1
```

```
For j = 1 To i - 1
```

```
If num(j) > num(j + 1) Then
```

```
tmp = num(j + 1)
```

```
num(j + 1) = num(j)
```

```
num(j) = tmp
```

```
End If
```

```
Next j
```

```
Next i
```

```
Close #1
```

```
For i = 1 To n
```

```
Print num(i);
```

```
If i Mod 10 = 0 Then Print
```

```
Next i
```

```
End Sub
```

上述过程先定义了一个空数组，接着打开数据文件 Data.dat(该文件存放在 D 盘 VBtemp 文件夹内)，读取第一个数据(50)，并用此值重定义数组的大小。接下来，For 循环从第二个数据开始，把 50 个数据读到数组 num 中，然后对这 50 个数值数据排序，并在窗体上输出排序结果，如图 8-3 所示。

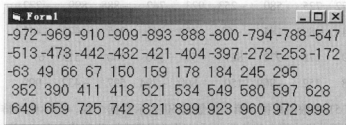


图 8-3 数值数据排序执行结果

2. Line Input #语句

格式如下：

Line Input # <文件号> , <字符串变量>

功能：从打开的文件中读取当前字符行，删除行末 <回车> <换行> 符后赋值给“字符串变量”。

说明：

- 1) 格式中的“文件号”的含义同前。
- 2) 格式中的“字符串变量”可以是一个字符串简单变量名，也可以是一个字符串数组元素名，用来接收从顺序文件中读出的字符行。
- 3) 它可以读取顺序文件中一行的全部字符，直至遇到回车符为止。此外，对于用 ASCII 码存放在磁盘上的各种语言源程序，都可以用 Line Input #语句一行一行地读取。
- 4) Line Input #语句与 Input #语句功能类似。只是 Input #语句读取的是文件中的数据项，而 Line Input #语句读的是文件中的一行。
- 5) Line Input #语句常用来复制文本文件。
- 6) Line Input #语句也可用于随机文件。

例如，设 STR 为字符串变量，#1 文件的当前行内容为“□□□101B□□"Franklin □□" Male□□18□□□ <回车> <换行>”，则执行 Line Input #1, STR 的结果如下：

```
STR = "□□□101B□□"Franklin□□" Male□□18□□□"
```

【例 8-6】把例 8-3 建立的学生数据文件 StudentInfo. dat 读到内存并在文本框中显示出，然后把文本框中的内容存入另一个磁盘文件 CopyStudentInfo. dat。

界面设计很简单，在窗体上建立一个文本框，在属性窗口中把该文本框的 MultiLine 属性设置为 True，Text 属性设置为“”。

在程序设计时，考虑到 Line Input #语句每次只能读取文件中的一行，所以必须使用循环。如何判断是否已经读取完全部数据呢？VB 提供了一个函数用于判断是否已到文件结尾，这个函数就是 EOF()。

该函数的格式如下：

EOF(文件号)

其功能是用来测试文件的结束状态。对于顺序文件来说，如果已经到文件末尾，则 EOF 函数返回 True，否则返回 False。EOF 函数常用来在循环中测试是否已到文件尾，一般结构如下：

Do While Not EOF(文件号)

文件读写语句

Loop

本例中的程序设计也采用了上述的结构，具体程序代码如下：

```
Private Sub Form_Click()
```

```
Dim str As String,txt As String
```

```
Open "d:\VBTemp\StudentInfo. dat" For Input As #1
```

```
Text1. FontSize = 12
```

```
Do While Not EOF(1)
```

```
Line Input #1, str
```

```
txt = txt & str & Chr(13) & Chr(10)
```

```
Loop
```

```
Text1. text = txt
```



```

Close #1
Open "d:\VBTemp\CopyStudentInfo.dat" For Output As #1
Print #1, Text1.text
Close #1
End Sub

```

上述过程首先打开一个磁盘文件 StudentInfo.dat, 用 Line Input#语句把该文件的内容一行一行地读到变量 str 中, 每读一行, 就把该行内容添加到变量 txt 中, 并加上回车换行符。然后把变量 txt 的内容放到文本框中, 并关闭该文件。此时, 文本框中显示文件 StudentInfo.dat 的内容, 如图 8-4 所示。之后, 程序建立一个名为 CopyStudentInfo.dat 的文件, 并把文本框的内容写入该文件。程序运行结束后, 文本框及两个磁盘文件中具有相同的内容。

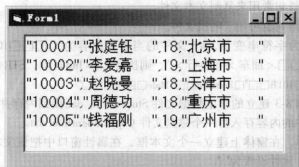


图 8-4 在文本框中显示文件内容

3. Input 函数

格式如下:

Input[\$] (<读取字符数>,[#]<文件号>)

功能: 从打开的文件中读取指定个数的字符, 包括单字节的西文字符、控制字符或双字节的中文字符。

说明:

- 1) 格式中的“文件号”的含义同前。
- 2) 格式中的“读取字符数”是一个数值表达式, 用以确定所读字符个数。
- 3) 该函数把一个文件作为非格式的字符流来读取, 不把回车-换行序列看做是一次输入操作的结束标志。因此, 当需要用程序从文件中读取单个字符时, 或者是用程序读取一个二进制的或非 ASCII 码文件时, 使用该函数较为方便。

【例 8-7】 利用 Input 函数, 把例 8-3 建立的学生数据文件 StudentInfo.dat 中的首条记录内容读到内存并在文本框中显示出来。

界面设计与例 8-6 相同。

程序代码如下:

```

Private Sub Form_Click()
    Dim str As String
    n = InputBox("请输入读取字符的个数:")

```

```

Open "d:\VBTemp\StudentInfo.dat" For Input As #1
Text1.FontSize = 12
str = Input(n, #1)
Text1.Text = str
Close #1
End Sub

```

上述过程首先询问读取字符的个数，然后打开磁盘文件 StudentInfo.dat，用 Input 函数读取相应个数的字符在文本框中加以显示。

根据例 8-6，已知文件 StudentInfo.dat 的首条记录对应为“10001”，“张庭钰”，“18”，“北京市”，因此可知包括“回车”“换行”在内为 36 个字符，不包括“回车”“换行”则为 34 个字符。由于“回车”“换行”为控制字符无法显示，因此在程序弹出对话框询问读取字符个数时，可以输入 34~36 区间的任意一个数，都可以将 StudentInfo.dat 文件中首条记录的内容进行显示，如图 8-5a 所示。如果想将第二条记录的学号部分“10002”也一并显示，则只需输入 36+7=43 个读取字符数即可，如图 8-5b 所示。由此可知，Input 函数是将文件作为非格式的字符流来读取的。

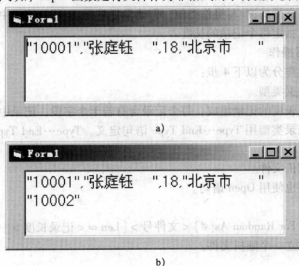


图 8-5 Input 函数读取字符流

a) 读取未含“回车”“换行”的字符流 b) 读取包含“回车”“换行”的字符流

8.3 随机文件

在随机文件中，构成文件的记录是定长的，记录之间无需添加分隔符以示区别。例如，文件 NameList：

```

00001 张峰000002 赵茵茵000003Tom...01000 东方致远

```

就构成了一个随机文件，每个记录长 $(5+10)$ 字节。

由于记录定长，可直接定位记录的开始和结束，如第 n 条记录从文件中开始位置 $(n-1) \times 15 + 1$ 字节处开始。因此，可任意读/写随机文件的某条记录。

随机文件具有如下特点：

1) 随机文件的记录是定长的，只有给出记录号 n ，才能通过 $(n-1) \times \text{记录长度} + 1$ 计算出该记录与文件首记录的相对地址。因此，在用 Open 语句打开文件时必须指定记录的长度。

2) 每个记录划分为若干个字段，每个字段的长度等于相应的变量的长度。

3) 各变量要按一定格式置入相应的字段。

4) 打开随机文件后，既可读也可写。

随机文件以记录为单位进行操作。本节中“记录”兼有两方面的含义，一个是指记录类型，即用 Type...End Type 语句定义的类型；另一个是指要处理的文件的记录。两者有联系，也有区别，要注意区分。

8.3.1 随机文件的读/写操作

随机文件与顺序文件的读/写操作类似，但通常把需要读/写的记录中的各字段放在一个记录类型中，同时应指定每个记录的长度。

1. 随机文件的写操作

随机文件的写操作分为以下 4 步：

第 1 步：定义数据类型。

随机文件由固定长度的记录组成，每个记录含有若干个字段。记录中的各字段可以放在一个记录类型中，记录类型用 Type...End Type 语句定义。Type...End Type 语句通常在标准模块中使用。

第 2 步：打开随机文件。

随机文件的打开也使用 Open 语句。

格式如下：

Open <文件名> For Random As [#]<文件号> [Len = <记录长度>]

功能：打开或建立一个随机文件。

说明：

1) 格式中的“文件号”的含义同前。

2) 与顺序文件不同，文件以随机访问方式打开后，可以同时写入和读出操作。

3) 格式中的“记录长度”等于各字段长度之和，以字符(字节)为单位。要指明“记录长度”，否则记录的默认长度为 128 个字节。

第 3 步：将内存中的数据写入磁盘。

随机文件的写操作通过 Put 语句来实现。

格式如下：

Put [#]<文件号>,[<记录号>],<数据项>

功能：将一个记录变量的内容写入所打开的磁盘文件中指定的记录位置处。

说明：

1) 格式中的“文件号”的含义同前。

2) 格式中的“记录号”是取值范围 $1 \sim 2^{31} - 1$ (即 $1 \sim 2147483647$) 的整数, 表示写入的是第几条记录。如果省略记录号, 则表示在当前记录后插入一条记录。省略记录号后, 不能省略逗号。例如 Put #1, , buf。

3) 格式中的“数据项”是一个表达式, 该表达式中可包含除对象变量和数组变量外的任何变量(包括含有单个数组元素的下标变量)。

4) 对随机文件, 若记录号 $= i (i \geq 1)$ 且 $\langle \text{数据项长度} \rangle \geq \langle \text{记录长度} \rangle$, 则数据将被输出在文件的第 $(i-1) \times \langle \text{记录长度} \rangle + 1$ 字节至第 $i \times \langle \text{记录长度} \rangle$ 字节。若 $\langle \text{数据项长度} \rangle > \langle \text{记录长度} \rangle$, 则发生“记录长度错误”; 若 $\langle \text{数据项长度} \rangle < \langle \text{记录长度} \rangle$, 则文件的第 $(i-1) \times \langle \text{记录长度} \rangle + 1 + \langle \text{数据项长度} \rangle$ 字节至第 $i \times \langle \text{记录长度} \rangle$ 字节内容不确定。一般地, $\langle \text{数据项长度} \rangle$ 应等于 Open 语句的 Len 子句所定的 $\langle \text{记录长度} \rangle$ 。

第4步: 关闭文件。

关闭文件的操作与顺序文件相同。

2. 随机文件的读操作

从随机文件中读数据与写数据的操作步骤类似, 只是将第3步中的 Put 语句用读数据的 Get 语句来代替即可。

格式如下:

Get # <文件号> , [<记录号>] , <变量>

功能: 将由“文件号”所指定的磁盘文件中的数据读到“变量”中。

说明:

1) 格式中的“文件号”的含义同前。

2) 格式中的“记录号”是取值范围 $1 \sim 2^{31} - 1$ (即 $1 \sim 2147483647$) 的整数, 表示读出的的是第几条记录。如果省略记录号, 则表示在当前记录后读出一条记录。省略记录号后, 不能省略逗号。例如 Get #1, , buf。

3) 格式中的“变量”可以是除对象变量和数组变量外的任何变量(包括含有单个数组元素的下标变量)。

4) 对随机文件, 若记录号 $= i (i \geq 1)$ 且 $\langle \text{变量所需数据长度} \rangle \geq \langle \text{记录长度} \rangle$, 则将读出文件的第 $(i-1) \times \langle \text{记录长度} \rangle + 1$ 字节至第 $i \times \langle \text{记录长度} \rangle$ 字节。若 $\langle \text{变量所需数据长度} \rangle > \langle \text{记录长度} \rangle$, 则发生“记录长度错误”; 若 $\langle \text{变量所需数据长度} \rangle < \langle \text{记录长度} \rangle$, 则文件的第 $(i-1) \times \langle \text{记录长度} \rangle + 1 + \langle \text{数据项长度} \rangle$ 字节至第 $i \times \langle \text{记录长度} \rangle$ 字节内容无用。一般地, $\langle \text{变量所需数据长度} \rangle$ 应等于 $\langle \text{记录长度} \rangle$ 。

下面通过例子说明随机文件的读/写操作。

【例8-8】建立一个随机存取的化妆品库存文件, 然后读取文件中的记录。

假设每条记录包含4个字段, 记录格式见表8-2。

表8-2 化妆品库存文件的记录格式

化妆品名称	生产厂家	库存数量	价 格
...

按以下步骤操作:



第1步：定义数据类型

化妆品库存文件的每个记录包含4个字段，各字段的长度(字节数)和数据类型见表8-3。

表8-3 字段的长度和数据类型

项 目	类 型	长 度	项 目	类 型	长 度
化妆品名称(Nam)	字符串	10	库存数量(Num)	整型	2
生产厂家(Unt)	字符串	15	价格(Pre)	单精度	4

根据以上定义的记录格式，可以在标准模块中对该记录类型进行如下定义：

```
Type Store
```

```
    Nam As String * 10
```

```
    Unt As String * 15
```

```
    Num As Integer
```

```
    Pre As Single
```

```
End Type
```

然后定义该类型的变量：

```
Public st As Store
```

第2步：打开随机文件，并指定记录长度。

由于随机文件的记录长度是固定的，因此应在打开文件时用 Len 语句来指定记录长度。如果不指定，则系统默认为 128 字节。从记录类型的结构可以计算出其长度为 $10 + 15 + 2 + 4 = 31$ 个字节。所以可使用下面的 Open 语句打开文件：

```
Open "d:\VBtemp\Store.dat" For Random As #1 Len = 31
```

其实，VB 提供了非常方便的计算记录长度的函数 Len()。其调用方法如下：

记录长度 = Len(记录类型变量)

则上面的 Open 语句可改写如下：

```
Open "d:\VBtemp\Store.dat" For Random As #1 Len = Len(st)
```

上面语句中有两个 Len，其中等号左边的 Len 是 Open 语句中的子句，而等号右边的 Len 是一个函数。

第3步：从键盘上输入记录中的各个字段，对文件进行读/写操作。

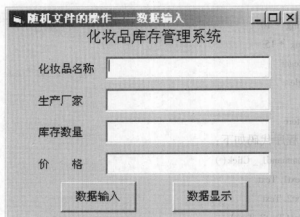
根据以上分析，可以为应用程序设计用户界面了。程序采用两个窗体，一个用于输入数据，另一个用于输出数据。两个窗体的界面设计如图 8-6a 和图 8-6b 所示。其中，图 8-6a 中的 4 个文本框由上至下依次为 Text1、Text2、Text3、Text4，两个命令按钮由左至右依次为 Command1、Command2；图 8-6b 中的文本框为 Text1，两个命令按钮由左至右依次为 Command1、Command2。

通过数据输入窗体的 4 个文本框将数据输入后，单击数据输入按钮，通过以下程序实现数据记录向磁盘文件的写入：

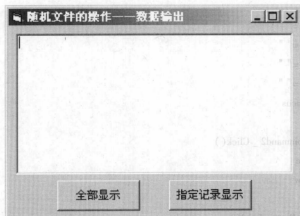
```
st.Nam = Text1.Text
```

```
st.Unt = Text2.Text
```

```
st.Num = Val(Text3.Text)
```



a)



b)

图 8-6 窗体界面设计

a) 数据输入窗体 b) 数据输出窗体

```
st.Prc = Val(Text4.Text)
```

```
Put #1,,st
```

用以上程序可以把一条记录写入磁盘文件 Store.dat，反复执行这段程序可以完成多条记录的输入。记录输入后，不必关闭文件，就可以从文件中读取记录，例如：

```
Get #1,2,st
```

以上语句读取文件中的第二条记录。

第4步：关闭文件。

```
Close #1
```

以上是建立和读取化妆品库存文件的一般操作，下面给出完整的程序。

标准模块中的程序代码如下：



```
Type Store
```

```
    Nam As String * 10
```

```
    Unt As String * 15
```

```
    Num As Integer
```

```
    Prc As Single
```

```
End Type
```

```
Public st As Store
```

数据输入窗体中程序代码如下:

```
Private Sub Command1_Click()
```

```
    st.Nam = Text1.Text
```

```
    st.Unt = Text2.Text
```

```
    st.Num = Val(Text3.Text)
```

```
    st.Prc = Val(Text4.Text)
```

```
    Put #1,,st
```

```
    Text1.Text = ""
```

```
    Text2.Text = ""
```

```
    Text3.Text = ""
```

```
    Text4.Text = ""
```

```
    Text1.SetFocus
```

```
End Sub
```

```
Private Sub Command2_Click()
```

```
    Form1.Hide
```

```
    Form2.Show
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
    Open "d:\VBtemp\Store.dat" For Random As #1 Len = Len(st)
```

```
End Sub
```

数据输出窗体程序代码如下:

```
Private Sub Command1_Click()
```

```
    Text1.Text = ""
```

```
    r_number = LOF(1)/Len(st)'计算文件中的记录数
```

```
    For i = 1 To r_number
```

```
        Get #1,i,st
```

```
        Text1.Text = Text1.Text & st.Nam
```

```
        Text1.Text = Text1.Text & Space(3)
```

```
        Text1.Text = Text1.Text & st.Unt
```

```
        Text1.Text = Text1.Text & Space(3)
```

```
        Text1.Text = Text1.Text & st.Num
```

```
        Text1.Text = Text1.Text & Space(3)
```

```

Text1.Text = Text1.Text & st.Prc
Text1.Text = Text1.Text & Chr(13)
Text1.Text = Text1.Text & Chr(10)
Next i
End Sub
Private Sub Command2_Click()
n = InputBox("请输入要显示的记录号:")
Get #1,n,st
Text1.Text = st.Nam
Text1.Text = Text1.Text & Space(3)
Text1.Text = Text1.Text & st.Unt
Text1.Text = Text1.Text & Space(3)
Text1.Text = Text1.Text & st.Num
Text1.Text = Text1.Text & Space(3)
Text1.Text = Text1.Text & st.Prc
End Sub
Private Sub Form_Unload(Cancel As Integer)
Close #1
End
End Sub

```

其中,数据输出窗体程序的 Command1_Click 中调用的函数 LOF() 是 VB 系统提供的返回文件字节数的函数。调用格式如下:

LOF(文件号)

功能: 返回给文件分配的字节数(即文件长度)。

可以使用该函数方便地计算出文件中的记录条数。

程序的执行情况如下:

1) 程序运行后,在数据输入窗体中键入一条记录内容,单击“数据输入”按钮,将记录内容写入磁盘文件 Store.dat。如此执行多次,将如表 8-4 中的数据输入文件。

表 8-4 程序中输入的数据

Nam	Unt	Num	Prc
金牌儿童霜	天津郁美净集团	100	21.5
儿童爽身粉	天津郁美净集团	200	7.5
大宝眼袋霜	大宝化妆品公司	150	7.13
眼角皱纹蜜	大宝化妆品公司	130	8.05
大宝磨砂膏	大宝化妆品公司	300	17.25

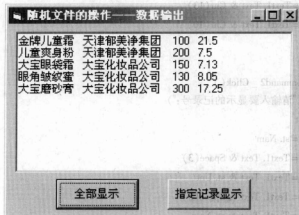
2) 单击“数据显示”按钮,打开数据输出窗体。

3) 单击数据输出窗体中的“全部显示”按钮,顺序读取 Store.dat 文件中的每一条记录并在文本框中显示出来,如图 8-7a 所示。

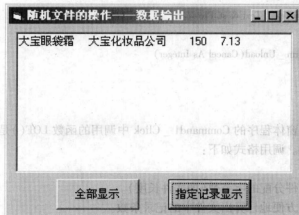
4) 单击数据输出窗体中的“指定记录显示”按钮,通过记录号读文件记录,在弹出的



对话框中键入 3 后, 在文本框中显示记录号为 3 的记录, 如图 8-7b 所示。



a)



b)

图 8-7 记录的显示

a) 显示全部记录 b) 显示指定记录

8.3.2 记录操作

随机文件又称为记录文件, 常用操作有删除记录、增加记录、替换记录。

1. 增加记录

对于随机文件, 增加记录实际上是将记录添加到文件尾部。增加记录的步骤如下:

- 1) 找到随机文件的记录数, 即最后一个记录的记录号, 将记录数加 1。
- 2) 输入需要增加的记录。
- 3) 将输入的记录用 Put 语句写到随机文件的尾部。
- 4) 如果还有记录需要增加, 重复上面的步骤。

例如, 例 8-8 中数据输入窗体中“数据输入”按钮对应的事件处理过程 Command1_Click 改为如下代码, 则每次键入记录内容后, 单击“数据输入”按钮将把记录内容添加到

随机文件的尾部，不论随机文件是新创建的还是打开的原来建立的文件。

```
Private Sub Command1_Click()  
    st.Nam = Text1.Text  
    st.Unt = Text2.Text  
    st.Num = Val(Text3.Text)  
    st.Pre = Val(Text4.Text)  
    r_number = LOF(1)/Len(st)'计算文件中的记录数  
    r_number = r_number + 1  
    Put #1, r_number, st  
    Text1.Text = "  
    Text2.Text = "  
    Text3.Text = "  
    Text4.Text = "  
    Text1.SetFocus  
End Sub
```

以上程序与例8-8中程序的区别在于每次用Put语句向磁盘文件写入记录之前，先求出文件中已有的记录数，然后根据该值将新添加的记录定位于文件尾部。用以上代码代替例8-8中的相应代码，运行程序，然后输入表8-5的数据。

表8-5 添加的记录内容

Nam	Unt	Num	Pre
雪晶活力霜	大宝化妆品公司	265	37.95
手足护理霜	大宝化妆品公司	170	6.10
雪肤活力蜜	大宝化妆品公司	350	9.20

输入结束后，单击“数据显示”按钮，打开数据输出窗体。单击数据输出窗体中的“全部显示”按钮，将显示文件中所有的记录，如图8-8所示。从中可以看出，新增加的记录已经附加在原来记录的后面。



图8-8 增加记录的显示

2. 替换记录

用新的记录代替原有的记录,其步骤如下:

- 1) 给出被替换的记录号。
- 2) 输入新记录。
- 3) 用 Put 语句将新记录按被替换的记录号写入随机文件。

例如,将例 8-8 中数据输入窗体中增加一个“替换记录”按钮,对应的事件处理过程 Command3_Click 的代码如下:

```
Private Sub Command3_Click()  
    r_position = Val( InputBox("请输入替换记录号:"))  
    r_number = LOF(1)/Len(st)'计算文件中的记录数  
    If r_position > r_number Or r_position < 1 Then  
        MsgBox "文件中不包含此记录!"  
        Exit Sub  
    End If  
    st.Nam = Text1.Text  
    st.Unt = Text2.Text  
    st.Num = Val( Text3.Text)  
    st.Pre = Val( Text4.Text)  
    Put #1, r_position, st  
    Text1.Text = ""  
    Text2.Text = ""  
    Text3.Text = ""  
    Text4.Text = ""  
    Text1.SetFocus  
End Sub
```

以上程序中,待用户键入新记录内容后,先要求用户输入欲替换的记录号,在判断记录号未引起越界的情况下,将新记录内容写入文件中旧记录的位置完成替换。改动后的例 8-8 数据输入窗体界面如图 8-9a 所示,运行程序,输入表 8-6 的数据替换文件中的第 3 条记录,然后打开数据输出窗体,单击“全部显示”按钮,将显示文件中的所有记录,如图 8-9b 所示。与图 8-8 相比,从中可以看出,新记录已经替换了原来的记录。

表 8-6 替换的记录内容

Nam	Unt	Num	Pre
靓肤精华液	天津郁美净集团	10	108

3. 删除记录

对于随机文件,删除记录只是将被删除的记录后面的记录依次向前移动一个记录位置,覆盖住需要删除的记录。记录数不改变,后面将会出现重复的记录。为了改变记录数,消除重复的记录,需要如下几点:

- 1) 生成一个新的随机文件。

随机文件的操作——数据输入

化妆品库存管理系统

化妆品名称

生产厂家

库存数量

价 格

数据输入 替换记录 数据显示

随机文件的操作——数据输出

金牌儿童霜	天津郁美净集团	100	21.5
儿童爽身粉	天津郁美净集团	200	7.5
靓肤精华液	天津郁美净集团	10	108
眼角皱纹蜜	大宝化妆品公司	130	8.05
大宝磨砂膏	大宝化妆品公司	300	17.25
雪晶活力霜	大宝化妆品公司	265	37.95
手足护理霜	大宝化妆品公司	170	6.1
雪肤活力蜜	大宝化妆品公司	350	9.2

全部显示 指定记录显示

b)

图 8-9 替换记录的显示

a) 数据输入窗体 b) 替换后的记录内容

- 除了到尾部的重复记录外，复制原文件的所有记录到新文件。
- 关闭这两个文件。
- 用 Kill 语句删除旧文件。

Kill 语句用于删除磁盘上的文件，其格式如下：

Kill <文件名>

其中，“文件名”是字符串表达式，用来指定要删除的文件的文件名(包括路径)。例如，Kill "d:\VBTemp\1.txt"将删除 D 盘下 VBTemp 文件夹内的 1.txt 文件。

当“文件名”指定的文件不存在时会发生错误。

- 用 Name 语句将新文件名改成原文件名。

Name 语句用于将文件改名，其格式如下：



Name < 原文件名 > As < 新文件名 >

其中:

1) Name 语句可以对文件或文件夹重命名,也可用来移动文件,但不能跨越驱动器移动文件。

2) 格式中的“原文件名”和“新文件名”都是字符串表达式,分别用来指定已存在和改名后的文件名(包括路径)。

3) “原文件名”与“新文件名”必须在同一驱动器上。

4) 如果“新文件名”指定的路径与“原文件名”指定的路径不同但文件名相同,则 Name 语句把文件移到新的文件夹下,且保持文件名不变。

5) 用 Name 语句可以移动文件,不能移动文件夹,但可以对文件夹重命名。

6) 当“原文件名”不存在,或者“新文件名”已存在时,都会发生错误。

例如,将 D 盘 VBtemp 文件夹内的 1. dat 文件重命名为 2. dat 文件,可使用如下语句:

```
Name "d:\VBtemp\1. dat" As "d:\VBtemp\2. dat"
```

由上述分析可知,删除记录较增加或替换记录操作起来要复杂一些。下面,仍然在例 8-8 的基础上加以改进,使之具备删除记录的能力。打开例 8-8 的工程文件,在数据输出窗体中添加一个“删除记录”按钮,其对应的事件处理过程 Command3_Click 的代码如下:

```
Private Sub Command3_Click()  
    r_position = Val(InputBox("请输入要删除的记录号:"))  
    r_number = LOF(1)/Len(st)'计算文件中的记录数  
    If r_position > r_number Or r_position < 1 Then  
        MsgBox "文件中不包含此记录!"  
        Exit Sub  
    End If  
    Do While r_position < r_number '该循环完成被删记录的覆盖和后续记录的前移  
        Get #1, r_position + 1, st  
        Put #1, r_position, st  
        r_position = r_position + 1  
    Loop  
    Open "d:\VBtemp\tmpStore. dat" For Random As #2 Len = Len(st)  
    For n = 1 To r_number - 1 '该循环完成文件的复制  
        Get #1, n, st  
        Put #2, n, st  
    Next n  
    Close #1  
    Close #2  
    Kill "d:\VBtemp\Store. dat" '删除原文件  
    Name "d:\VBtemp\tmpStore. dat" As "d:\VBtemp\Store. dat" '将新生成的文件重命名  
    Open "d:\VBtemp\Store. dat" For Random As #1 Len = Len(st) '重新打开文件  
    Command1_Click '调用全部显示按钮的事件处理过程显示删除后剩余的记录
```

End Sub

以上代码,首先询问欲删除的记录号,判断记录号未越界后,用一个 Do While...Loop 循环完成被删除记录的覆盖与后续记录的前移。之后新建一个名为 tmpStore.dat 的临时文件,用一个 For 循环将原文件中除最后一条记录(重复的那条多余的记录)之外的其余全部记录依次读出并写入新建的临时文件中。关闭两个文件,用 Kill 语句删除原来的文件,并用 Name 语句将临时文件重命名为原文件的名称 Store.dat。最后,将该 Store.dat 文件再次打开,并调用“全部显示”按钮的事件处理过程将删除后剩余的记录全部显示。运行以上程序,打开数据输出窗体,先单击“全部显示”按钮,在文本框中显示 Store.dat 文件中的全部记录,如图 8-10a 所示。然后单击“删除记录”按钮,在弹出的对话框中输入 3,程序将完成对文件中第 3 条记录的删除,删除后文件的内容将在文本框中加以显示,如图 8-10b 所示。从图中可见,第 3 条记录被删除了。

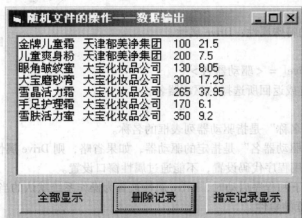
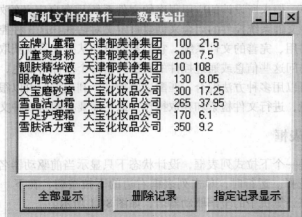


图 8-10 删除记录的显示

a) 记录删除之前的文件内容 b) 记录删除之后的文件内容

8.4 文件系统控件

计算机的文件系统包括系统软件、应用软件中的文件及用户建立的数据文件。为使程序设计人员在设计的程序中能够使用文件系统, VB 提供了两种选择, 既可以使用由通用对话框(CommonDialog)控件提供的 Open 和 Save As 标准对话框, 也可以使用驱动器列表框(Drive List Box)、目录列表框(Directory List Box)和文件列表框(File List Box)这 3 种文件系统控件的组合创建自定义对话框。

CommonDialog 控件提供了标准的 Save As 和 Open 对话框, 所谓标准就是它与其他 Windows 应用程序的同类对话框相同, 具有标准化的外观, 并且能够识别可用的网络驱动器。因此, 如果应用程序只需要具备保存、打开文件的功能, 则应使用 CommonDialog 控件。

但在某些情况下, 要求所设计的应用程序的文件系统具有自定义的外观或特殊的功能, 这就要使用上述 3 种文件系统控件的组合来创建自定义对话框。这 3 种控件都经过了特别设计, 提供了简单、易用、完善的文件系统, 它们都能自动从操作系统获取与文件系统相关的信息。程序员可以访问这些信息或通过控件的属性判断每个控件的信息, 可以单独使用某个文件系统控件, 也可以用多种方法混合、匹配这些控件。利用它们可以很容易地建立起一个可视的图形操作环境, 进行文件检索和存取操作。以下简要介绍这 3 种文件系统控件。

8.4.1 驱动器列表框

驱动器列表框是一个下拉式列表框, 设计状态下只显示当前驱动器名。其主要作用是实现驱动器的可视化选择。运行时, 单击列表框右边的下拉按钮, 在下拉窗口中可以看到该计算机中的全部驱动器名, 如图 8-11 所示。

1. 驱动器列表框的属性

驱动器列表框的基本属性: Name、Width、Height、Left、Top、Enabled、Visible 等。

驱动器列表框的特殊属性: Drive 属性。

格式如下:

[< 名称 > .] Drive [= < 驱动器名 >]

功能: 用来设置或返回所选择的驱动器名。

说明:

- 1) 格式中的“名称”是指驱动器列表框的名称。
- 2) 格式中的“驱动器名”是指定的驱动器, 如果省略, 则 Drive 属性是当前驱动器。
- 3) 该属性只能用程序代码设置, 不能通过属性窗口设置。
- 4) 在设计状态下和程序启动时, Drive 属性被置为操作系统默认的当前驱动器。

2. 驱动器列表框的事件

驱动器列表框最常用的事件是 Change 事件。每次重新设置 Drive 属性, 都会引发 Change 事件。驱动器列表框的默认名称为 Drive1, 其 Change 事件过程的名称为 Drive1_Change()。



图 8-11 驱动器列表

8.4.2 目录列表框

目录列表框用来显示当前驱动器上的目录结构，它是一种下拉框列表结构。刚建立时显示当前驱动器的顶层目录和当前目录。顶层目录用打开的文件夹表示，当前目录用阴影的文件夹表示，当前目录下的子目录用关闭的文件夹来表示，如图 8-12 所示。

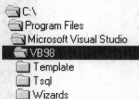


图 8-12 目录列表框

1. 目录列表框的属性

目录列表框的基本属性：Name、Width、Height、Left、Top、Enabled、Visible 等。

目录列表框的特殊属性：Path 属性。

格式如下：

[<名称> .]Path[= <路径名>]

功能：用来设置或返回当前驱动器的路径。

说明：

- 1) 格式中的“名称”是指目录列表框的名称。
- 2) 格式中的“路径名”是指定的路径，如果省略，则 Path 属性是当前路径。“路径名”的格式如下：

驱动器名:\文件夹名\...

- 3) 该属性只能用程序代码设置，不能通过属性窗口设置。

例如：

Print Dir1.Path 显示当前路径(Dir1 是目录列表框的默认名称)。

Dir1.Path = "e:\VBtemp\Practice" 将重新设置路径，目录列表框中显示 E 盘上 VBtemp 下的 Practice 目录下的目录结构。

2. 目录列表框的事件

改变目录列表框 Path 属性的值会激活 Change 事件。利用该事件可以改变路径或选择文件夹。

3. 目录列表框与驱动器列表框的同步

目录列表框与驱动器列表框有密切的联系。通常，改变驱动器列表框中的驱动器名后，目录列表框中的目录应当随之变为该驱动器上的目录，即目录列表框与驱动器列表框应连锁互动。如何达到连锁互动的效果呢？当改变驱动器列表框的 Drive 属性时，将产生 Change 事件。因此只要将 Drive1_Change 事件过程中，将 Drive1.Drive 的属性赋给 Dir1.Path，就可以产生连锁互动的效果。其程序代码如下：

```
Private Sub Drive1_Change()
```

```
Dir1.Path = Drive1.Drive
```

```
End Sub
```

当在窗体上放置一个驱动器列表框和一个目录列表框，并执行上面的程序代码时，将显示图 8-13 所示的运行结果。

8.4.3 文件列表框

驱动器列表框和目录列表框控件可以用来指定当前驱动器和当前目录，而文件列表框可

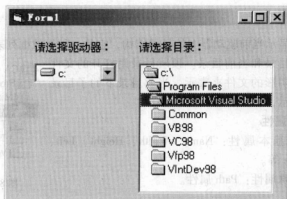


图 8-13 目录列表框与驱动器列表框的同步

以用于显示当前目录下的文件。

文件列表框的默认控件名为 File1。

1. 文件列表框的主要属性

(1) FileName 属性

该属性表示在文件列表框中显示的文件，且该属性的设置可以是一个文件名，也可以是按文件类型划分的扩展名(如 *.txt)，或是所有文件 (*.*) 等。同样，该属性不能在设计时设置，只能在运行时在事件过程中用程序代码实现。

(2) Pattern 属性

该属性用来设置在执行时要显示的某一种类型的文件，可以在设计阶段用属性窗口设置，也可以通过程序代码设置。在默认情况下，Pattern 的属性值为 “*.*”。在程序代码中设置 Pattern 的格式如下：

文件列表框 . Pattern[= 属性值]

2. 文件列表框与驱动器列表框、目录列表框的同步

在实际应用中，驱动器列表框、目录列表框和文件列表框往往需要同步动作，这可以通过目录列表框 Path 属性的改变引发 Change 事件来实现。其程序代码如下：

```
Private Sub Dir1_Change()  
    File1.Path = Dir1.Path  
End Sub
```

该事件过程使窗体上的目录列表框 Dir1 和文件列表框 File1 实现同步。当目录列表框的 Path 属性改变时将触发 Dir1_Change 事件过程，所以在 Dir1_Change 事件过程中，把 Dir1.Path 赋给 File1.Path，则目录列表框就可与文件列表框产生同步。

同样，加上下面的程序，就可起到 3 个文件系统控件连锁互动的作用，运行结果如图 8-14 所示。

```
Private Sub Drive1_Change()  
    Dir1.Path = Drive1.Drive  
End Sub
```



图 8-14 驱动器、目录与文件列表框的同步

8.5 文件操作应用

8.5.1 传统的文件处理机制

前面介绍的使用 Open 语句以及其他相关语句和函数对文件的处理方法属于传统的文件处理机制，这些机制在 VB6.0 中仍得到完全支持。

【例 8-9】设计一个文件管理程序，在文件列表框中双击某个图形文件时，在图像框中显示该图形文件；在文件列表框中双击某个纯文本文件时，在文本框中显示该文本文件；在文件列表框中双击某个可执行文件时，将运行该程序。文件管理程序界面如图 8-15 所示。

驱动器列表框用于选定当前的驱动器名，目录列表框用于选定当前的文件路径，文件列表框用于文件的选择。组合框用于选定在文件列表框内所显示的文件类型。例如，当选中“JPEG 文件 (*.jpg)”选项时，则在文件列表框中显示当前路径下所有扩展名为“.jpg”的文件，双击文件列表框中扩展名为“.jpg”的文件，该图形文件就显示在图像框中；当选中“文本文件 (*.txt)”选项时，则在文件列表框中显示当前路径下所有扩展名为“.txt”的文件，双击文件列表框中扩展名为“.txt”的文件，该文本文件就显示在文本框中；当选中“可执行文件 (*.exe)”选项时，则在文件列表框中显示当前路径下所有扩展名为“.exe”的文件，双击文件列表框中扩展名为“.exe”的文件，就可运行该程序；当选中“所有文件 (*.*)”选项时，则在文件列表框中显示当前路径下所有的文件。

通过组合框过滤文件列表框中显示的内容，是利用文件列表框的 Pattern 属性实现。通过组合框的 ListIndex 属性，可以区分当前被选中的内容，据此来设置 Pattern 属性的值。

对于 jpg 文件的显示，是利用图像框实现的，将图像框的 Stretch 属性设置为 True，以显示全部图形文件。使用 LoadPicture 函数为图像框的 Picture 属性加载图像文件。

对于 txt 文件的显示，是利用文本框实现的，将文本框的 MultiLine 属性设置为 True，允

许多行显示；同时将文本框的 ScrollBars 属性设置为 3，允许水平、垂直方向均有滚动条。对文本文件操作时，先用 Open 语句打开文件，然后用 Line Input 语句按行读出文件内容，添加到文本框内，最后用 Close 语句关闭文件。

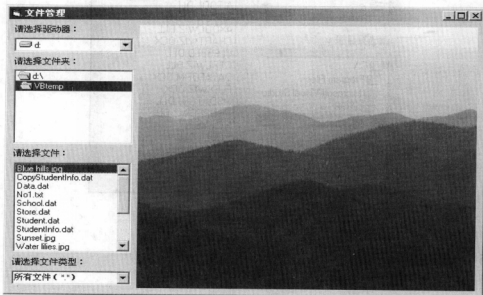


图 8-15 文件管理程序界面

操作图像文件和文本文件时，其文件路径名由目录列表框的 Path 属性和文件列表框的 FileName 属性构造而成。

图像框和文本框不能同时出现，根据用户的选择，需要显示图形文件时，令图像框的 Visible 属性为 True，文本框的 Visible 属性为 False；需要显示文本文件时，设置相反。

对于“.exe”文件的执行，是使用 Shell 函数完成的。

程序代码如下：

```
Private Sub Combo1_Click()
```

```
    Select Case Combo1.ListIndex
```

```
        Case 0
```

```
            File1.Pattern = "*.jpg"
```

```
        Case 1
```

```
            File1.Pattern = "*.txt"
```

```
        Case 2
```

```
            File1.Pattern = "*.exe"
```

```
        Case 3
```

```
            File1.Pattern = "*.*"
```

```
    End Select
```

```
End Sub
```

```
Private Sub Dir1_Change()
```

```
File1.Path = Dir1.Path
```

```
End Sub
```

```
Private Sub Drive1_Change()
```

```
Dir1.Path = Drive1.Drive
```

```
End Sub
```

```
Private Sub File1_DblClick()
```

```
Dim filetype As String
```

```
Filetype = UCase(Right(File1.FileName,3))
```

```
Select Case filetype
```

```
Case "JPG"
```

```
Text1.Visible = False
```

```
Image1.Visible = True
```

```
Image1.Picture = LoadPicture(Dir1.Path & "\ " & File1.FileName)
```

```
Case "TXT"
```

```
Image1.Visible = False
```

```
Text1.Visible = True
```

```
Open Dir1.Path & "\ " & File1.FileName For Input As #1
```

```
Text1.FontSize = 12
```

```
Do While Not EOF(1)
```

```
Line Input #1, iStr
```

```
txt = txt & iStr & Chr(13) & Chr(10)
```

```
Loop
```

```
Text1.Text = txt
```

```
Close #1
```

```
Case "EXE"
```

```
Shell File1.FileName
```

```
End Select
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
Text1.Left = 2505
```

```
Text1.Top = 105
```

```
Image1.Left = Text1.Left
```

```
Image1.Top = Text1.Top
```

```
Image1.Width = Text1.Width
```

```
Image1.Height = Text1.Height
```

```
Image1.Stretch = True
```

```
Text1.Visible = False
```

```
Combo1.AddItem "JPEG 文件(*.jpg)"
```

```

Combo1.AddItem "文本文件( *.txt)"
Combo1.AddItem "可执行文件( *.exe)"
Combo1.AddItem "所有文件( *, *)"
Combo1.ListIndex = 3
End Sub

```

8.5.2 文件系统对象处理机制

文件系统对象(File System Object,简称 FSO 对象)模型是 VB6.0 新增加的一项功能,用于对文件系统进行管理。它通过一种基于对象的方式,使用户在编写程序时可以通过这些对象提供的丰富的属性和方法来操作和管理计算机的文件系统。

1. 文件系统对象概述

FSO 对象模型提供了一组对文件系统的驱动器、文件夹和文件进行管理的对象,主要包括以下几个对象:

1) FileSystemObject 对象:是文件系统对象中的主要对象,能提供一整套用于创建、删除、收集相关信息和通常操作的驱动器、文件夹和文件的方法,很多与本对象关联的方法是复制的其他对象的。

2) File 对象:文件对象,提供了创建、删除或移动文件等操作,并能向系统查询文件的路径和名称等信息。

3) Folder 对象:文件夹对象,允许对文件夹进行创建、删除或移动操作,也能向系统查询文件夹的路径和名称等。

4) Drive 对象:驱动器对象,它可以提供诸如驱动器的可用空间大小和共享名等信息。这里的“驱动器”既可以是一个硬盘上的逻辑硬盘符,也可以是 CD-ROM 驱动器、RAM 驱动器以及网络映射驱动器等。

5) TextStream 对象:文本流对象,允许对文本文件进行读和写操作。

2. 使用文件系统对象编程

FSO 对象是 Scripting 类库中的对象而非 VB 的内部对象,Scripting 类库位于 Windows 文件夹内 System 文件夹中的 Sccrun.Dll 文件里,使用它之前必须确保在工程中引用了这个文件。方法是执行“工程”菜单的“引用”命令,打开“引用对话框”选择 Microsoft Scripting Runtime 复选框。在添加完引用后即可使用文件系统对象了。这时,会在“对象浏览器”中添加一个新的“Scripting”类库,可以在其中查看它的对象、集合、方法、事件和常数等等。

使用 FSO 对象模型编程的主要步骤:

(1) 创建文件系统对象

要使用文件系统对象编程首先应该创建文件系统对象,以便于对它进行相应的处理。可以通过如下两种方法完成:

1) 使用关键字 New,将一个变量声明为 FileSystemObject 对象类型,例如:

```
Dim fso As New FileSystemObject
```

2) 使用 CreateObject 函数创建一个 FileSystemObject 对象,例如:

```
Set fso = CreateObject("Scripting.FileSystemObject")
```

在上面的例子中, Scripting 是类型库的名称, FileSystemObject 则是想要创建一个实例的对象的名称。

(2) 使用 FileSystemObject 对象的方法

创建完了 FileSystemObject 对象后, 就可以使用该 FileSystemObject 对象的适当的方法去实现相应创建、删除、复制和移动文件或文件夹等的基本操作。

1) 创建对象的方法: CreateFolder 方法或 CreateTextFile 方法(FSO 对象模型不支持创建或删除驱动器)。

2) 删除对象方法: FileSystemObject 对象的 DeleteFile 和 DeleteFolder 方法, 或者 File 对象和 Folder 对象的 Delete 方法。

用户根据实际情况使用适当的方法, 还可以复制、移动文件和文件夹。FileSystemObject 对象模型中有一些功能是冗余的, 用户可以灵活使用。

3) 访问已有的驱动器、文件和文件夹。用户如果想访问一个已存在的驱动器、文件或文件夹, 应该使用 FileSystemObject 对象中相应的“get”方法: GetDrive、GetFolder 和 GetFile。例如:

```
Dim fso As New FileSystemObject, fl As File
Set fl = fso.GetFile("d:\VBtemp\No1.txt")
```

对新创建的对象不需要使用“get”方法, 因为 Create 函数已经返回了一个句柄到新创建的对象。例如, 如果使用 CreateFolder 方法创建了一个新的文件夹, 就没有必要使用 GetFolder 方法来访问该对象的诸如名称、路径、大小等属性。只要给 CreateFolder 函数设置一个变量来获取新建文件夹的句柄, 就可以访问其属性、方法和事件。其程序代码如下:

```
Dim fso As New FileSystemObject, fldr As Folder
Set fldr = fso.CreateFolder("c:\aaa")
MsgBox "新建的文件夹是:" & fldr.Name
```

(3) 访问对象的属性

一旦有了对象的句柄, 就能够访问其属性。例如, 要获得一个特定文件夹的名称, 首先要创建该对象的一个实例, 然后通过适当的方法(在本例中是 GetFolder 方法, 因为该文件夹已经存在)得到其句柄:

```
Set fdr = fso.GetFolder("d:\VBtemp")
```

得到了 Folder 对象的句柄, 就可以查看其属性了, 比如查看 Name 属性:

```
MsgBox "文件夹名称为:" & fdr.Name
```

如果想要找出一个文件的最新修改时间, 可以使用如下的语法:

```
Dim fso As New FileSystemObject, fl As File
Set fl = fso.GetFile("d:\VBtemp\No1.txt") '获得要查询的 File 对象
MsgBox "文件最后被访问时间:" & fl.DateLastAccessed '显示信息
```

3. 驱动器管理

在 FSO 对象模型中, Drive 对象主要用于管理驱动器。通过调用 Drive 的属性既可以读取系统本地驱动器的信息, 又可以得到网络驱动器的信息。在编程时, 通过对 FileSystemObject 对象使用 GetDrive 方法, 建立一个 Drive 对象的实例, 然后调用 Drive 对象的属性以获取驱动器的信息。Drive 对象的属性见表 8-7。



表 8-7 Drive 对象的属性

属 性	描 述
AvailableSpace	以字节表示的驱动器或网络上的用户可用的磁盘空间
DriveType	驱动器类型的值(0~5)。其中, 0 表示 Unknown, 1 表示 Removable, 2 表示 Fixed, 3 表示 Network, 4 表示 CD-ROM, 5 表示 RAMDisk
DriveLetter	本地驱动器或网络共享的驱动器符号
FileSystem	驱动器使用的文件系统类型, 诸如 FAT、FAT32、NTFS 等
FreeSpace	以字节表示的驱动器或网络上的用户可用的磁盘剩余空间
IsReady	驱动器可用否, 如果指定驱动器可用, 该值为 True, 否则为 False
Path	指定文件、文件夹或驱动器的路径
RootFolder	返回一个 Folder 对象, 表示一个指定驱动器的根文件夹
SerialNumber	用于唯一标识磁盘卷标的十进制数字序列号
ShareName	网络共享名
TotalSize	以字节表示的驱动器或网络共享的总空间大小
VolumeName	设置或返回指定驱动器的卷标名

【例 8-10】 利用 Drive 对象的属性获取驱动器的信息, 如图 8-16 所示。

程序代码如下:

```
Private Sub Command1_Click()  
    ' 创建一个 FileSystemObject 对象 fso  
    Dim fso As New FileSystemObject  
    ' 将变量 drv 声明为 Drive 对象类型  
    Dim drv As Drive, s As String  
    Set drv = fso.GetDrive("C:") ' 返回一个 Drive 对象  
    ' 调用 Drive 对象的属性获得驱动器信息  
    s = "驱动器" & ("C:") & vbCrLf  
    s = s & "磁盘总容量:" & FormatNumber(drv.TotalSize/1024, 0)  
    s = s & " Kb" & vbCrLf  
    s = s & "剩余空间:" & FormatNumber(drv.FreeSpace/1024, 0)  
    s = s & " Kb" & vbCrLf  
    MsgBox s, , "Drive 对象"  
End Sub
```

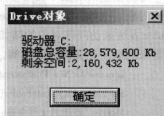


图 8-16 获取驱动器的信息

本例首先创建一个 FileSystemObject 对象, 对其使用 GetDrive 方法生成一个 Drive 对象, 调用该 Drive 对象的属性获得了驱动器的信息。FormatNumber 函数用于对表达式的值进行格式化, 参数中的 0 表示保留 0 位小数。

4. 文件夹管理

使用 FSO 对象模型对文件夹的管理包括文件的创建、删除、移动、复制以及检索相关的文件夹的信息。FileSystemObject 对象和 Folder 对象均能完成对文件夹的管理工作, 但使用

Folder 对象的属性还可以获取文件夹的信息, Folder 对象的属性见表 8-8。表 8-9 列出了文件夹管理可以使用的部分方法。

表 8-8 Folder 对象的属性

属 性	描 述
Attributes	设置或返回文件夹的属性(如, I 为只读)
DateCreated	返回指定文件夹的创建日期和时间
DateLastAccessed	返回最后一次访问文件夹的日期和时间
DateLastModified	返回最后一次修改文件夹的日期和时间
Drive	返回指定文件夹所在的驱动器符号
Files	返回文件夹中包含的文件的集合
IsRootFolder	返回指定文件夹是否为根文件夹(True 表示是)
Name	设置或返回文件夹的名称
ParentFolder	返回父文件夹名称
Path	返回文件夹的路径名
Size	返回以字节表示的包含在文件夹中的所有文件和子文件夹的大小
SubFolders	返回包含在文件夹中的子文件夹的集合

表 8-9 文件夹管理可以使用的部分方法

任 务	FileSystemObject 方法	Folder 方法
创建一个文件夹	CreateFolder	
删除一个文件夹	DeleteFolder	Delete
移动一个文件夹	MoveFolder	Move
复制一个文件夹	CopyFolder	Copy
获得当前文件夹的完整路径名称	GetAbsolutePathName	
查找一个文件夹是否在驱动器上	FolderExists	
获得已有 Folder 对象的一个实例	GetFolder	
找出一个文件夹的父文件夹的名称	GetParentFolderName	
找出系统文件夹的路径	GetSpecialFolder	

(1) 文件夹的创建

FileSystemObject 对象的 CreateFolder 方法用于建立一个文件夹, 其格式如下:

Object.CreateFolder (foldername)

其中, foldername 是以字符串表达式形式描述的要创建的文件夹, 如果指定的文件夹已经存在, 则发生错误。

例如, 以下语句将在 C 盘上分别建立两个名为 big 和 small 的文件夹。

```
Dim fso As New Scripting.FileSystemObject
fso.CreateFolder ("c:\big")
fso.CreateFolder ("c:\small")
```




(2) 文件夹的复制、移动、删除

要完成文件夹的复制、移动、删除操作，既可以使用 FileSystemObject 对象的相关方法，也可以使用 Folder 对象的相关方法。

例如，将上例创建的 small 文件夹移动到 big 文件夹中，可以用两种方法实现。

方法 1：使用 FileSystemObject 对象的相关方法。程序代码如下：

```
Dim fso As New Scripting.FileSystemObject
```

```
fso.MoveFolder "c:\small", "c:\big\small"
```

方法 2：使用 Folder 对象的相关方法。程序代码如下：

```
Dim fso As New Scripting.FileSystemObject, fdr As Folder
```

```
Set fdr = fso.GetFolder("c:\small")
```

```
fdr.Move "c:\big\small"
```

(3) 获取与文件夹有关的信息

FileSystemObject 对象和 Folder 对象配合使用可以获取关于文件夹的全部信息。编程时，通过 FileSystemObject 对象建立一个 Folder 对象实例，然后调用 Folder 对象的相关属性获取文件夹的信息。

例如，以下代码用于查看前面建立的 big 文件夹的部分属性：

```
Dim fso As New Scripting.FileSystemObject, fdr As Folder
```

```
Dim s As String
```

```
Set fdr = fso.GetFolder("c:\big")
```

```
'获得文件夹名和文件夹所在的驱动器
```

```
s = "文件夹" & fdr.Name & "在" & UCase(fdr.Drive) & _
```

```
"驱动器上" & vbCrLf & vbCrLf
```

```
'获得最后一次访问时间
```

```
s = s & "最后一次访问时间：" & fdr.DateLastAccessed
```

```
MsgBox s
```

5. 文件管理

使用 FSO 对象模型对文件的管理包括文件的创建(打开)、删除、移动、复制以及检索相关的文件信息。FileSystemObject 对象和 File 对象均能完成对文件的管理工作，但使用 File 对象的属性还可以获取文件的信息，File 对象的属性见表 8-10。表 8-11 列出了文件管理可以使用的部分方法。

表 8-10 File 对象的属性

属 性	描 述
Attributes	设置或返回文件的属性(如,1 为只读)
DateCreated	返回指定文件的创建日期和时间
DateLastAccessed	返回最后一次访问文件的日期和时间
DateLastModified	返回最后一次修改文件的日期和时间
Drive	返回指定文件所在的驱动器符号
Name	设置或返回指定文件的名称

(续)

属 性	描 述
ParentFolder	返回所在文件夹的名称
Path	返回所在文件夹的路径
Size	返回以字节表示的包含在文件夹中的所有文件和子文件夹的大小
Type	返回文件的类型描述

表 8-11 文件管理可以使用的部分方法

任 务	FileSystemObject 方法	File 方法
创建(打开)一个文件	CreateTextFile 或 OpenTextFile	OpenAsTextStream
删除一个文件	DeleteFile	Delete
移动一个文件	MoveFile	Move
复制一个文件	CopyFile	Copy
获得当前文件的完整路径名称	GetAbsolutePathName	
查找一个文件是否在驱动器上	FileExists	
获得已有 File 对象的一个实例	GetFile	
从一个路径描述中获取文件名	GetFileName	
返回随机产生的文件名字符串	GetTempName	

(1) 文件的创建与打开

打开文件或创建文件有 3 种不同方法,具体操作如下:

1) 使用 FileSystemObject 对象的 CreateTextFile 方法。

格式如下:

<对象名>.CreateTextFile(<文件名>[,<覆盖否>,[Unicode]])

功能:创建一个指定文件名的文件,并返回一个用于对该文件读写的 TextStream 对象。

说明:

<对象名>:是一个 FileSystemObject 对象的名字。

<文件名>:字符串表达式,表示新创建文件的文件名。

<覆盖否>:若为 True,新创建的文件将覆盖原文件,默认为 False。当该参数为 False 时,如果创建的文件已经存在,则发生错误,所以应使用 FileExists 方法提前判断文件是否存在。

Unicode:若为 True,创建 Unicode 文件;若为 False,创建 ASCII 文件;默认为 False。

要创建一个空文本文件,可以使用以下语句:

```
Dim fso As New FileSystemObject, tso
Set tso = fso.CreateTextFile("d:\vbtemp\fso.txt", True)
Set tso = fso.GetFile("d:\vbtemp\fso.txt")
```

注意:FSO 对象模型尚不支持创建随机文件或二进制文件。要创建随机文件和二进制文件,请使用带 Random 或 Binary 标志的 Open 命令。

【例 8-11】 创建一个名为 fsotest.txt 的文件，并在文件中写入“FileSystemObject 对象”。程序代码如下：

```
Private Sub Command1_Click()  
    Dim fso,tso  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    If fso.FileExists("d:\vbtemp\fsotest.txt") Then '判断文件是否存在  
        MsgBox "文件已存在!"  
        Unload Me  
    Else  
        Set tso = fso.CreateTextFile("d:\vbtemp\fsotest.txt",True)  
        tso.WriteLine("FileSystemObject 对象") '向文件中写  
        tso.Close  
    End If  
End Sub
```

运行此例后，可在 Windows 下用记事本打开 fsotest.txt 文件查看结果。

2) 使用 FileSystemObject 对象的 OpenTextFile 方法。

格式如下：

<对象名>.OpenTextFile(<文件名>[,<方式>[,<创建否>[,<文件格式>]])

功能：打开一个指定的文件并返回一个 TextStream 对象，该对象可用于对文件进行读操作或追加操作。

说明：

<对象名>：是一个 FileSystemObject 对象的名字。

<文件名>：字符串表达式，表示新创建或打开的文件的文件名。

<方式>：表示输入/输出方式。可为以下 3 个常数之一：ForReading、ForAppending 或 ForWriting。

<创建否>：表示如果指定的文件名不存在是否可以创建一个新文件。设置为 True，表示创建新文件；设置为 False，表示不创建文件。默认值为 False。

<文件格式>：表示打开文件的格式。设置为 True，表示以 Unicode 格式打开文件；设置为 False，表示以 ASCII 格式打开。默认值为 False。

【例 8-12】 创建一个名为 fsotest1.txt 的文件，并在文件中写入“FileSystemObject 对象”。

程序代码如下：

```
Private Sub Command1_Click()  
    Dim fso As New FileSystemObject,tso  
    Set tso = fso.OpenTextFile("d:\vbtemp\fsotest1.txt",ForWriting,True)  
    tso.Write "FileSystemObject 对象" '向文件中写  
    tso.Close  
End Sub
```

3) 使用 File 对象的 OpenAsTextStream 方法。

格式如下:

<对象名>: OpenAsTextStream([<方式>[,<文件格式>]])

功能: 打开一个指定的文件并返回一个 TextStream 对象, 该对象可用于对文件进行读、写、追加操作。

说明:

<对象名>: 是一个 File 对象的名字。

<方式>: 表示输入/输出方式。可为以下 3 个常数之一: ForReading、ForAppending 或 ForWriting。

<文件格式>: 表示打开文件的格式。如果省略, 表示以 ASCII 格式打开。

【例 8-13】 用写方式创建一个名为 fsotest2.txt 的文件, 并在文件中写入“FileSystemObject 对象”。

程序代码如下:

```
Private Sub Command1_Click()  
    Dim fso, fo, tso  
    Set fso = CreateObject("Scripting.FileSystemObject")  
    fso.CreateTextFile "d:\vbtemp\fsotest2.txt" '创建一个文件  
    Set fo = fso.GetFile("d:\vbtemp\fsotest2.txt") '获得一个 File 对象的文件句柄  
    Set tso = fo.OpenAsTextStream(2) '创建 TextStream 对象用于写  
    tso.Write "FileSystemObject 对象"  
    tso.Close  
End Sub
```

(2) 文件的读/写

在文件打开或新创建后, 才能使用 TextStream 对象进行文件的读/写操作。TextStream 对象与文件读/写操作有关的方法见表 8-12。

表 8-12 TextStream 对象与文件读/写操作有关的方法

方 法	描 述
Read(n)	从文件中读取 n 个字符并返回得到的字符串
ReadAll	读取整个的 TextStream 文件并返回得到的字符串
ReadLine	从文件中读一行(不包括换行符)并返回得到的字符串
Write(String)	将字符串 String 写入文件中
WriteBlankLines(n)	将 n 个换行符写入文件
WriteLine(String)	将字符串 String 写入文件并在行尾加上换行符

【例 8-14】 创建一个名为 fsotest3.txt 的文件, 在文件中写入“FileSystemObject 对象”, 然后再将信息读出显示。

程序代码如下:

```
Private Sub Command1_Click()  
    Dim fso, fo, tso, sString  
    Set fso = CreateObject("Scripting.FileSystemObject")
```

```

fso.CreateTextFile "d:\vbtemp\fsotest3.txt" '创建一个文件
Set fo = fso.GetFile("d:\vbtemp\fsotest3.txt") '获得一个 File 对象的文件句柄
Set tso = fo.OpenAsTextStream(2) '创建 TextStream 对象用于写
'在文件中写一个字符串
tso.Write "FileSystemObject 对象"
tso.Close
Set tso = fo.OpenAsTextStream()
sString = tso.ReadLine '读一行
tso.Close
MsgBox sString '显示读出的信息
End Sub

```

(3) 文件的关闭

关闭由 TextStream 对象打开的文件用 Close 方法。例如 tso.Close。

(4) 文件信息的获取

FileSystemObject 对象和 File 对象必须配合使用才能获取文件的有关信息。

【例 8-15】 执行以下程序可以查询由文本框输入的程序名称所对应程序的相关信息。

运行时由文本框输入 c:\windows\system32\calc.exe, 显示计算器程序的相关信息, 如图 8-17 所示。

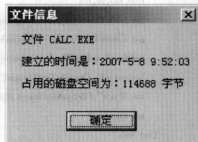
程序代码如下:

```

Private Sub Command1_Click()
    Dim fso, fo, stemp
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set fo = fso.GetFile(Text1.Text) '获得一个 File 对象
    sString = "文件" & UCase(fo.Name) & vbCrLf & vbCrLf '获得文件名
    '获得文件创建时间
    sString = sString & "建立的时间是:" & fo.DateCreated & vbCrLf & vbCrLf
    '获得文件的大小
    sString = sString & "占用的磁盘空间为:" & fo.Size & "字节"
    MsgBox sString, "文件信息" '显示获得的文件信息
End Sub

```

图 8-17 获取文件信息



(5) 文件的删除、移动与复制

FileSystemObject 对象和 File 对象均提供了对文件进行删除、移动与复制的方法。

1) 使用 FileSystemObject 对象的 DeleteFile、MoveFile、CopyFile 方法。

【例 8-16】 执行以下程序将 fsotest.txt 文件由 D 盘根目录移至 D 盘下 vbtemp 文件夹内。运行该程序前, 应在 D 盘根目录建立 fsotest.txt 文件和 vbtemp 文件夹。

```

Private Sub Command1_Click()

```

```

Dim fso
Set fso = CreateObject("Scripting.FileSystemObject")
fso.MoveFile "d:\fsotest.txt", "d:\vbtemp\"

```

End Sub

2) 使用 File 对象的 Delete、Move、Copy 方法。

【例 8-17】 执行以下程序将 fsotest1.txt 文件由 D 盘根目录复制到 D 盘下 vbtemp 文件夹内。运行该程序前，应在 D 盘根目录建立 fsotest1.txt 文件和 vbtemp 文件夹。

```

Private Sub Command1_Click()
Dim fso, fo
Set fso = CreateObject("Scripting.FileSystemObject")
Set fo = fso.GetFile("d:\fsotest1.txt")'获得 File 对象句柄
fo.Copy "d:\vbtemp\"
End Sub

```

6. FSO 应用实例

【例 8-18】 设计界面如图 8-18 所示的程序，实现从源地址到目标地址文件和文件夹的复制和移动。

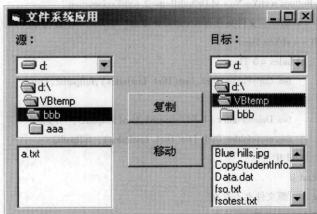


图 8-18 文件系统应用界面

程序代码如下：

```

Dim fso, ofolder, ofile
Private Sub Command1_Click(Index As Integer)
' 获得带"\的完整路径名
If Mid(Dir1.List(Dir1.ListIndex), 4, 1) <> "\" Then
    fullpath1 = Dir1.List(Dir1.ListIndex) & "\"
Else
    fullpath1 = Dir1.List(Dir1.ListIndex)
End If
If Mid(Dir2.List(Dir2.ListIndex), 4, 1) <> "\" Then

```

```

        fullpath2 = Dir2. List( Dir2. ListIndex) & "\"
Else
    fullpath2 = Dir2. List( Dir2. ListIndex)
End If

If File1. FileName = "" Then ' 没有选择文件! 复制目录
    If Right( fullpath1, 2) = ":\" Then
        MsgBox "不能进行整盘复制, 请选择文件夹"
    Else
        ' 下列 5 行语句要获取源目录列表框中不带路径名的当前目录名称
        i = 2
        Do While Left( Right( fullpath1, i), 1) <> "\"
            i = i + 1
        Loop
        fullpathtemp = Right( fullpath1, i - 1)
        fullpath3 = fullpath2 & fullpathtemp
        If fso. FolderExists( fullpath3) Then ' 源文件夹已存在
            sButtons = vbYesNo + vbDefaultButton2 + vbExclamation
            s = MsgBox("文件夹已存在, 是否要覆盖?", sButtons, "警告")
            If s = vbYes Then
                If Index = 0 Then
                    fso. CopyFolder Dir1. List( Dir1. ListIndex), fullpath2, True
                Else
                    fso. DeleteFolder ( Left( fullpath3, Len( fullpath3) - 1))
                    fso. movefolder Dir1. List( Dir1. ListIndex), fullpath2
                End If
            End If
        End If
    Else ' 源文件夹不存在
        If Index = 0 Then
            fso. CopyFolder Dir1. List( Dir1. ListIndex), fullpath2, True
        Else
            fso. movefolder Dir1. List( Dir1. ListIndex), fullpath2
        End If
    End If
End If

Else ' 复制文件
    fullfile2 = fullpath2 & File1. FileName
    If fso. FileExists( fullfile2) Then
        sButtons = vbYesNo + vbDefaultButton2 + vbExclamation
        s = MsgBox("文件已存在, 是否要覆盖?", sButtons, "警告")
    End If
End If

```

```

If s = vbYes Then '源文件已存在
    s1 = fullpath1 & File1.FileName
    If Index = 0 Then
        fso.CopyFile s1, fullpath2, True
    Else
        fso.DeleteFile (fullfile2)
        fso.MoveFile s1, fullpath2
    End If
End If
Else '源文件不存在
    s1 = fullpath1 & File1.FileName
    If Index = 0 Then
        fso.CopyFile s1, fullpath2, True
    Else
        fso.MoveFile s1, fullpath2
    End If
End If
End If
File2.Refresh
Dir2.Refresh
File1.Refresh
End Sub
Private Sub Command2_Click(Index As Integer)
End Sub
Private Sub Form_Load()
Set fso = CreateObject("Scripting.FileSystemObject")
Drive1.Drive = "c:\\" '初始化源驱动器列表框
Drive2.Drive = "c:\\" '初始化目标驱动器列表框
End Sub
Private Sub Drive1_Change()
'使驱动器列表框和目录列表框同步
Dir1.Path = Drive1.Drive
ChDrive Drive1.Drive '使系统的当前驱动器变为 drive1.Drive 所指定的驱动器
ChDir "c:\\" '使默认的系统目录为"c:\\"
End Sub
Private Sub Dir1_Change()
'使目录列表框和文件列表框同步
File1.Path = Dir1.Path
ChDir Dir1.Path '使系统的当前目录变为 dir1.Path 所指定的目录

```




```
End Sub  
Private Sub Drive2_Change()  
    Dir2.Path = Drive2.Drive  
    ChDrive Drive2.Drive  
End Sub  
Private Sub Dir2_Change()  
    File2.Path = Dir2.Path  
    ChDir Dir2.Path  
End Sub
```

其中, ChDrive 和 ChDir 语句用于更改系统的当前驱动器和目录。程序中使用它们是为了使文件系统的当前驱动器、当前目录与用户通过文件系统控件所选择的驱动器和目录保持一致。例如:

```
ChDrive Drive1.Drive
```

使系统的当前驱动器变为 Drive1.Drive 所指定的驱动器。

习 题

- 8-1 将第7章例7-17的简易CAD程序加以改造,使之可以保存用户的绘图内容,实现反复编辑。
- 8-2 通过窗体界面输入每个职工的工号、姓名、性别、年龄、籍贯,单击“确定”按钮将职工信息存入一个顺序文件,该文件的保存位置及文件名可以任意指定。
- 8-3 打开上题建立的文件,在输入某职工的工号后找出与之相关的信息,并将查找结果显示在窗体上。
- 8-4 用FSO对象模型设计一个可以向文本文件中添加学生成绩的程序,文件结构采用定长结构,包括学生的学号、姓名、语文成绩、数学成绩、外语成绩。统计学生的平均成绩,然后再按照分数由高到低依次排序,生成一个新文件。
- 8-5 用FSO对象模型实现例8-9。

第9章 网络和多媒体

随着网络技术和多媒体技术的出现,极大地促进了计算机技术的发展和信息技术的普及。基于网络技术的程序设计和多媒体程序在程序设计领域占有越来越重要的地位。这一章,将为大家介绍 VB 对于这两种技术的支持。

9.1 网络概述

所谓计算机网络,就是采用通信手段,将地理位置分散的、各自具备自主功能的若干台计算机有机地连接起来组成的一个复合系统,这个复合系统可以用来实现通信交往、资源共享和协同工作等目标。

按信息传输技术分类,可把网络分为广播式网络和点到点网络,前者在网络中只有单一的一个通信信道。按地理区域范围分类,可把网络分为局域网 LAN、城域网 MAN 和广域网 WAN,3 者跨越的范围依次增大。

Internet 是全球性的互联网络,它是由计算机和网络互相连接组成的庞大集合,任何一台 Internet 中的计算机都能够和网中其他计算机交换信息。

目前 Internet 上提供的较为流行的服务有:

- 全球范围的超媒体信息浏览服务(WWW);
- 远程登录(Telnet);
- 文件传输(FTP);
- 电子邮件(Email)。

其中,WWW 服务是目前广泛在 Internet 上使用的高级浏览服务,把存放于全球范围内的众多计算机上的信息以超媒体的方式链接在一起,构成了一个世界范围的网,称为 WWW (World Wide Web)。它制定了一套标准化且易懂的超文本描述语言(HTML 语言)、超文本传输协议(HTTP)和统一的资源定位格式(URL)。URL 规定了一个文档在 WWW 中存放地点的统一格式及地址。例如,http://www.tjut.edu.cn 就是一个 URL 地址,其中第一部分 http 表示所遵循的协议,第二部分就是信息资源所在的计算机名,它的后面还可以有目录名和文件名,如果省略,就表示主页。在进行 WWW 浏览时,通过浏览器看到的即是一个 HTML 格式的文件,它可以引导访问者进一步访问其他的信息和资源。

计算机网络节点之间要做到有条不紊的交换数据,每个节点必须遵守一些事先约定好的规则,这些规则对数据交换时的交换顺序、数据格式、流量控制等进行了一系列的规定,这些规定就是网络协议。网络协议采用层次结构加以组织,这样灵活性高,易于实现和维护,各层可以采用最合理的技术来实现,并且有利于促进标准化的发展。计算机网络层次结构模型和各层协议的集合称为计算机网络体系结构。至今,影响最大的是 ISO/OSI 参考模型和 TCP/IP 参考模型。

由国际标准化组织(ISO)发布的 OSI (Open System Interconnection,开放系统互联)参考模



型将通信会话需要的各种进程划分成 7 个相对独立的功能层次,即物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。其中,前 3 层提供了网络访问,后 4 层用于支持端端通信。TCP/IP 参考模型是 Internet 事实上的基础,目前被广泛地支持和应用,它是一个 4 层协议系统,包括链路层、网络层、传输层和应用层。

常用的 Internet 服务程序大都工作在应用层,使用的协议主要有 FTP、HTTP、Telnet、SMTP 等。其中 HTTP(Hypertext Transfer Protocol,超文本传输协议)是 Web 操作的基础,它是一个使信息能通过 Web 交换的客户/服务器协议。HTTP 定义了浏览器能提出的请求的类型以及服务器返回的响应的类型。通过 HTTP,用户可以从远程服务器中获取网页,或如果他有权限,就可以将网页存储在服务器中。HTTP 还提供向网页上添加新信息或全部删除它们的能力。FTP(File Transfer Protocol,文件传输协议)是进行多种格式文件访问的最常用协议,同 HTTP 一样,在互联网上有许多的 FTP 服务器,使用 FTP 进行工作,大多用来传递文件和发布文件等。Telnet 协议是 Internet 远程登录服务的标准协议,它为用户提供了在本地计算机上完成远程主机工作的能力。通过 Telnet 协议用户可以登录 Internet 上任何一台支持该协议的服务器,将本机虚拟为服务器的一个终端以使用其资源。SMTP(Simple Mail Transfer Protocol,简单邮件传输协议)是控制电子邮件从客户机传输到服务器或从某一个服务器传输到另一个服务器使用的传输协议。

在传输层中,最常用的是 TCP(Transmission Control Protocol,传输控制协议)和 UDP(User Datagram Protocol,用户数据报协议),这两个协议都是基于网络层的 IP(Internet Protocol,网际协议)。IP 是最底层的协议,它定义了数据按照数据报(Datagram)传输的格式和规则。TCP 通过序列确认以及包重发机制,提供可靠的数据流发送和到应用程序的虚拟连接服务。UDP 提供面向事务的简单的不可靠的信息传送服务。

为了区分连接在 Internet 上的主机,给每台联网主机都分配了一个专门的地址,称为 IP 地址。IP 地址是将计算机连接到 Internet 的网际协议地址,由 4B 组成,共 32bit,如 1100101001110001010010000000110。为便于记忆,将 IP 地址的 32bit 二进制数分成 4 段,每段 8bit,中间用小数点隔开,然后将每 8 位二进制数转换成十进制数,这样上述计算机的 IP 地址就变成了 202.113.72.6,这种格式被称做“点分十进制”格式。IP 地址的 4B 划分为两个部分,一部分用以标明具体的网络段,即网络标识(NetID);另一部分用以标明具体的节点,即主机标识(HostID)。同一个物理网络上的所有主机都用同一个网络标识,网络上的一个主机(包括网络上工作站、服务器和路由器等)都有一个主机标识与其对应。例如,前述 IP 地址其网络标识为 202.113.72,主机标识为 6。按照网络规模的大小,把 32bit IP 地址中用于网络标识和主机标识的位数设定成 3 种划分方式,这 3 种划分方式分别对应于 A 类、B 类、C 类 IP 地址。A 类 IP 地址由 1B 网络地址和 3B 主机地址组成,网络地址的最高位必须是“0”,其用于网络标识的长度为 7bit,主机标识的长度为 24bit。A 类网络地址数量较少,可以用于主机数达 1600 多万台的大型网络。B 类 IP 地址由 2B 网络地址和 2B 主机地址组成,网络地址的最高位必须是“10”,其用于网络标识的长度为 14bit,主机标识的长度为 16bit。B 类网络地址适用于中等规模的网络,每个网络所能容纳的计算机数为 6 万多台。C 类 IP 地址由 3B 的网络地址和 1B 主机地址组成,网络地址的最高位必须是“110”,其用于网络标识的长度为 21bit,主机标识的长度为 8bit。C 类网络地址数量较多,适用于小规模的网络,每个网络最多只能包含 254 台计算机。

同一台计算机上可以同时运行多个网络应用程序,在信息传输时如何加以区分呢?TCP和UDP采用16bit的“端口号”对应用程序加以识别,以确保目的地机器上的软件程序能从源地址机器处获得数据包,以及源计算机能收到正确的回复。端口号的范围为0~65535。有些端口号固定分配给某种服务,比如21端口分配给FTP服务,25端口分配给SMTP服务,80端口分配给HTTP服务等。

IP地址标识了连接在互联网上的一台计算机,端口号标识了计算机内的一个进程,将两者结合起来就可以唯一地标识网络上任意一台计算机内的任意一个网络程序。这个唯一的标识,称做“套接字”或“端点”。

客户/服务器模式是当前主流的网络模式。所谓客户/服务器模式,就是客户系统发出请求,服务器系统接收和处理请求,它们这种进行请求和处理的模式就叫做客户/服务器模式。

9.2 网络控件

VB6.0提供了3个ActiveX控件用来开发Internet应用程序,这3个控件分别是Winsock控件、Internet Transfer控件和WebBrowser控件。它们为用户提供了几乎每一层的Internet通信功能。

利用Winsock控件能开发基于TCP和UDP的Internet应用程序,并提供对TCP和UDP的低层访问,主要可用于创建计算机之间基于这两种协议的直接数据传输,如网络聊天程序。

利用Internet Transfer控件能开发FTP浏览和文件传输应用程序。该控件允许用户和WWW服务器、FTP服务器和Gopher服务器进行文件传输,以及采用同步或异步的方式浏览文件目录和检索数据。

WebBrowser控件集成了Microsoft Internet Explorer的大部分功能,可以利用这个控件方便地把IE浏览器的功能添加到一个应用程序中去。

以上3个控件都是ActiveX控件而非VB内部控件,使用它们之前必须确保在工程中添加了它们。方法是执行“工程”菜单的“部件”命令,打开“部件对话框”,选择Microsoft Winsock Control6.0(即Winsock控件)、Microsoft Internet Transfer Control6.0(即Internet Transfer控件)和Microsoft Internet Control(即WebBrowser控件)。

限于篇幅,本节只介绍Winsock控件的使用。

9.2.1 Winsock 控件的属性、方法和事件

1. Winsock 控件的属性

Winsock控件主要包括如下属性:

(1) LocalHostName 属性

功能:返回本地机器名,在设计时是只读的,而且是不可用的。

返回值类型:String。

(2) LocalIP 属性

功能:返回本地机器的IP地址,格式为xxx.xxx.xxx.xxx。该属性在设计时是只读的,

而且是不可用的。

返回值类型: String。

(3) LocalPort 属性

功能: 返回或者设置所用到的本地端口。在设计时是可读/写的, 而且是可用的。使用该属性时应知道:

对于客户端计算机, 该属性指定发送数据的本地端口。如果应用程序不需要特定端口, 则指定 0 为端口号。在这种情况下, 控件将选择一个随机端口。在建立起连接之后, 这就是用于 TCP 连接的本地端口。

对于服务器端计算机, 该属性指定用于侦听的本地端口。如果指定的是端口 0, 就使用一个随机端口。在调用了 Listen 方法后, 该属性就是已选定的实际端口号。

返回值类型: Long。

说明: 在计算机之间常用端口 0 来动态地建立连接。例如, 一个客户端希望服务器端给它回应, 它就可使用端口 0 获得新的(随机)端口号, 然后将该端口号交给远程计算机, 从而达到目的。

(4) Protocol 属性

功能: 返回或设置 Winsock 控件所使用的协议, 可以是 TCP, 或者是 UDP。Winsock 控件 Protocol 属性的设置值见表 9-1。

表 9-1 Winsock 控件 Protocol 属性的设置值

属 性 值	常 量	说 明	属 性 值	常 量	说 明
0	sckTCPProtocol	对应 TCP(默认值)	1	sckUDPProtocol	对应 UDP

说明: 在能够重新设置该属性之前必须(用 Close 方法)关闭控件。

(5) RemoteHost 属性

功能: 返回或设置远程计算机, 控件向它发送数据或从它那里接收数据。既可提供主机名, 比如 <http://www.tjut.edu.cn/>, 也可提供点格式下的 IP 地址字符串, 比如 202.113.66.66。

说明: 在指定该属性时, 应更新 URL 属性来显示新值。如果更新 URL 的主机部分, 则也要更新该属性来反映新值。

(6) RemoteHostIP 属性

功能: 返回远程机器的 IP 地址。使用该属性时应知道: 对于客户应用程序来说, 已经用 Connect 方法建立连接后, 该属性就包含了远程机器的 IP 字符串。对于服务器应用程序来说, 在请求连接(ConnectionRequest 事件)之后, 该属性包含远程计算机的 IP 字符串, 该字符串启动了连接。

当使用 UDP 时, 在 DataArrival 事件出现之后, 该属性包含了发送 UDP 数据的计算机的 IP 地址。

返回值类型: String。

(7) RemotePort 属性

功能: 返回或设置要连接的远程端口号。

返回值类型: Long。

说明: 在设置 Protocol 属性时, 将针对不同的高层协议自动把 RemotePort 属性设置成适

当的默认端口。例如, HTTP 对应 80 端口, FTP 对应 21 端口。

(8) SocketHandle 属性

功能: 返回一个与套接字句柄对应的值, 控件用套接字句柄同 Winsock 层通信。在设计时是只读的, 而且是不可用的。

返回值类型: Long。

说明: 该属性是为了传递到 WinsockAPI 而设计的。

(9) State 属性

功能: 返回控件的状态, 用枚举类型来表示, Winsock 控件 State 属性的设置值见表9-2。在设计时是只读的, 而且是不可用的。

表 9-2 Winsock 控件 State 属性的设置值

属性值	常 量	说 明	属性值	常 量	说 明
0	sckClosed	关闭 (默认值)	5	sckHostResolved	已识别主机
1	sckOpen	打开	6	sckConnecting	正在连接
2	sckListening	侦听	7	sckConnected	已连接
3	sckConnectionPending	连接挂起	8	sckClosing	同级人员正在关闭连接
4	sckResolvingHost	识别主机	9	sckError	错误

返回值类型: Integer。

2. WinSock 控件的方法

WinSock 控件包括如下方法:

(1) Accept 方法

格式如下:

对象名.Accept requestID

功能: 仅用于 TCP 服务器应用程序。在处理 ConnectionRequest 事件时用这个方法接受新连接。

说明:

- 1) 对象名: Winsock 控件对象的名称(下同)。
- 2) 在 ConnectionRequest 事件中使用 Accept 方法。ConnectionRequest 事件有一个对应的参数, 即 RequestID 参数, 该参数应该传给 Accept 方法。请看下例:

```
Private Sub Winsock1_ConnectionRequest (ByVal RequestID As Long)
```

```
    '测试 State 属性, 如果当前连接是打开的话, 则关闭连接
```

```
    If Winsock1.State <> sckClosed Then Winsock1.Close
```

```
    '将 RequestID 参数值传递给 Accept 方法
```

```
    Winsock1.Accept RequestID
```

```
End Sub
```

3) 应该在新的控件实例而不是侦听状态下的实例中使用 Accept 方法。这是因为, 一个 Winsock 控件只能接受一个连接请求, 而服务器应用程序多数情况下应能同时接受多个用户的连接。解决办法是创建 Winsock 控件数组, 动态地根据连接请求创建新的 Winsock 控件,

用新建的 Winsock 控件来接受新的请求，而此前用于侦听的 Winsock 控件仍保持侦听。因此，说明 2) 中的代码不要在需要应答多个客户端的应用中使用。

(2) Bind 方法

格式如下：

对象名 . Bind LocalPort, LocalIP

功能：指定用于 TCP 连接的 LocalPort 和 LocalIP。如果一台计算机上有多个网卡，就可使用这个方法。

说明：

1) LocalPort：该参数是 Long 类型，用于指定建立连接的端口。

2) LocalIP：该参数是 String 类型，用于指定建立连接的本地 Internet 地址。如果一台计算机上有多个网卡，可以用该参数来指定使用哪一个网卡。如果忽略该参数，控件使用的将是计算机上“控制面板”设置中“网络”控制面板对话框中列出的第一个网卡。

3) 在调用 Listen 方法之前必须调用 Bind 方法。

例如，本机有两块网卡，IP 地址分别为 192.168.1.100 和 192.168.1.105。则下列语句将使 Winsock1 控件侦听由第一块网卡接入的请求：

```
Winsock1.Bind 1088, "192.168.1.100"
```

```
Winsock1.Listen
```

(3) Close 方法

格式如下：

对象名 . Close

功能：对客户机和服务器应用程序关闭 TCP 连接或侦听套接字。

(4) Connect 方法

格式如下：

对象名 . Connect RemoteHost, RemotePort

功能：向远程计算机发起连接。

说明：

1) RemoteHost：要连接的远程计算机的名称。

2) RemotePort：要连接的远程计算机的端口。

3) 在想建立 TCP 连接时，必须调用 Connect 方法。

(5) GetData 方法

格式如下：

对象名 . GetData data, [type,][maxLen]

功能：获取当前的数据块并将其存储在变体类型的变量中。

说明：

1) data：GetData 方法成功执行之后，所获取的数据将存储在该参数中。如果对请求的类型没有足够可用的数据，则将 data 设置成 Empty。

2) type：可选项，用于指示获取数据的数据类型，可使用的设置值见表 9-3。默认值为 vbArray + vbByte。

表 9-3 数据类型常量

常 量	说 明	常 量	说 明
vbByte	Byte	vbDate	Date
vbInteger	Integer	vbBoolean	Boolean
vbLong	Long	vbError	SCODE
vbSingle	Single	vbString	String
vbDouble	Double	vbArray + vbByte	Byte Array(默认值)
vbCurrency	Currency		

3) **maxLen**: 可选项, 仅当 **type** 参数是字节数组或字符串时, 需要该参数, 用以指示接收的字节数组或字符串的大小。当 **type** 参数不是字节数组或字符串时, 则忽略这个参数。

4) 通常总是将 **GetData** 方法与 **DataArrival** 事件并用, 而 **DataArrival** 事件包含 **bytesTotal** 参数。如果指定一个比 **bytesTotal** 参数小的 **maxLen**, 则将得到警告 10040, 以此指出剩余的字节将丢失。

以下代码在 Winsock 控件的 **DataArrival** 事件中使用 **GetData** 方法获取数据, 并将数据存储在字符串变量中, 然后在文本框 **Text1** 中加以显示。

```
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
    Winsock1.GetData strData, vbString
    Text1.Text = Text1.Text & strData
End Sub
```

(6) Listen 方法

格式如下:

对象名. Listen

功能: 创建套接字并将其设置为侦听模式。该方法仅适用于 TCP 连接。

说明: 当有新连接时就会出现 **ConnectionRequest** 事件。处理 **ConnectionRequest** 事件时, 应用程序应该(在一个新的 Winsock 控件实例上)用 **Accept** 方法接受连接。

(7) PeekData 方法

格式如下:

对象名. PeekData data, [type,] [maxLen]

功能: 获取但不删除输入队列中的数据, 该方法仅适用于 TCP 连接。

说明: 该方法除了不删除输入队列中的数据之外, 其余均与 **GetData** 方法相同。

(8) SendData 方法

格式如下:

对象名. SendData data

功能: 将数据发送给远程计算机。

说明: 参数 **data** 是要发送的数据, 对于二进制数据应使用字节数组。

3. Winsock 控件的事件

Winsock 控件包括如下事件:

(1) Close 事件

格式如下:

对象名_ Close()

出现时机: 当远程计算机关闭连接时出现该事件, 在事件过程中应使用 Close 方法关闭 TCP 连接。

(2) Connect 事件

格式如下:

对象名_ Connect()

出现时机: 当一个 Connect 操作完成时发生, 使用该事件确认已经成功建立了一个连接。

(3) ConnectionRequest 事件

格式如下:

对象名_ ConnectionRequest(RequestID As Long)

出现时机: 当远程计算机请求连接时出现。

说明:

1) RequestID: 新连接请求标识, 应把此参数传递给下一个 Winsock 控件实例上的 Accept 方法。

2) 该事件仅适用于 TCP 服务器应用程序, 在客户程序请求一个新连接时激活该事件。激活事件之后, RemoteHostIP 和 RemotePort 属性中存储有客户端的信息。

3) 服务器可决定是否接受连接。如果不接受新连接, 则同级人员(客户)将得到 Close 事件。如果接受新连接, 则应在一个新控件实例上用 Accept 方法接受新连接。

(4) DataArrival 事件

格式如下:

对象名_ DataArrival (bytesTotal As Long)

出现时机: 当新数据到达时出现。

说明:

1) bytesTotal: 该参数数据类型为 Long, 指示了可获取的数据总数量。
2) 如果没有获取一个 GetData 调用中的全部数据, 则事件不会出现。只有存在新数据时才激活事件。可随时用 BytesReceived 属性检查可用的数据量。

(5) Error 事件

格式如下:

对象名_ Error(number As Integer, Description As String, Scode As Long, Source As String, HelpFile as String, HelpContext As Long, CancelDisplay As Boolean)

出现时机: 无论何时, 只要后台处理中出现错误(例如, 连接失败, 或者在后台收发数据失败)事件就会出现。

说明:

- 1) number: 定义错误代码的整数, Winsock 控件错误代码见表 9-4。
- 2) Description: 包含错误信息的字符串。
- 3) Scode: 表示长 SCODE。

表 9-4 Winsock 控件错误代码

值	常 量	说 明
7	sckOutOfMemory	内存不足
380	sckInvalidPropertyValue	属性值无效
394	sckGetNotSupported	属性不可读
383	sckSetNotSupported	属性是只读的
40006	sckBadState	所请求的事务或请求本身的错误协议或者错误连接状态
40014	sckInvalidArg	传递给函数的参数格式不确定, 或者不在指定范围内
40017	sckSuccess	成功
40018	sckUnsupported	不受支持的变量类型
40020	sckInvalidOp	在当前状态下的无效操作
40021	sckOutOfRange	参数越界
40026	sckWrongProtocol	所请求的事务或请求本身的错误协议
1004	sckOpCanceled	取消操作
10014	sckInvalidArgument	所请求的地址是广播地址, 但未设置标记
10035	sckWouldBlock	套接字不成块, 而指定操作将使之成块
10036	sckInProgress	制造块的 Winsock 操作在进行之中
10037	sckAlreadyComplete	完成操作。未进行制造块的操作
10038	sckNotSocket	描述符不是套接字
10040	sckMsgTooBig	数据报太大, 不适于缓冲区的要求, 因而被截断
10043	sckPortNotSupported	不支持指定的端口
10048	sckAddressInUse	地址在使用中
10049	sckAddressNotAvailable	来自本地机器的不可用地址
10050	sckNetworkSubsystemFailed	网络子系统失败
10051	sckNetworkUnreachable	此时不能从主机到达网络
10052	sckNetReset	在设置 SO_KEEPAIVE 时连接超时
11053	sckConnectAborted	由于超时或者其他失败而中止连接
10054	sckConnectionReset	通过远端重新设置连接
10055	sckNoBufferSpace	没有可用的缓冲空间
10056	sckAlreadyConnected	已连接套接字
10057	sckNotConnected	未连接套接字
10058	sckSocketShutdown	已关闭套接字
10060	sckTimeout	已关闭套接字
10061	sckConnectionRefused	强行拒绝连接
10093	sckNotInitialized	应首先调用 WinsockInit
11001	sckHostNotFound	授权应答: 未找到主机
11002	sckHostNotFoundTryAgain	非授权应答: 未找到主机
11003	sckNonRecoverableError	不可恢复的错误
11004	sckNoData	无效名, 对所请求的类型无数据记录

4) Source: 描述错误来源的字符串。

5) HelpFile: 包含帮助文件名的字符串。

6) HelpContext: Help 文件上下文。

7) CancelDisplay: 指示是否取消显示。默认值为 False, 显示默认的错误信息框; 如果不想使用默认的信息框, 则将 CancelDisplay 设置成 True。

(6) SendComplete 事件

格式如下:

对象名_SendComplete()

出现时机: 在完成一个发送操作时出现。

(7) SendProgress 事件

格式如下:

对象名_SendProgress (bytesSent As Long, bytesRemaining As Long)

出现时机: 在发送数据期间出现。

说明:

1) bytesSent: 从上次激活事件以来已发送的字节数。

2) bytesRemaining: 在发送缓冲区等待发送时的字节数。

9.2.2 Winsock 控件的使用

1. 通信协议的选择与设置

在使用 WinSock 控件时, 首先需要考虑使用什么协议。可以使用的协议包括 TCP 和 UDP。两种协议之间的重要区别在于它们的连接状态:

TCP 协议控件是基于连接的协议, 在开始数据传输之前, 用户必须先建立连接。

UDP 协议是一种无连接协议, 两台计算机之间传输消息时, 消息从一台计算机发送到另一台计算机, 但是两者之间没有明确的连接。另外, 单次传输的最大数据量取决于具体的网络。

到底选择哪一种协议通常是由需要创建的应用程序决定的。以下几点参考有助于选择适宜的协议:

1) 在收发数据的时候, 应用程序是否需要得到客户端或者服务器的确认信息? 如果需要, 使用 TCP, 在收发数据之前先建立明确的连接。

2) 数据量是否特别大(例如图像与声音文件)? 在连接建立之后, TCP 将维护连接并确保数据的完整性。不过, 这种连接需要更多的计算机资源。

3) 数据发送是间歇的, 还是在会话内? 例如, 如果应用程序在某个任务完成的时候需要通知某个计算机, UDP 是更适宜的。UDP 适合发送少量的数据。

选择好协议后, 需要对协议进行设置。在设计状态时, 可以在“属性”窗口中单击“协议”, 然后选择 `sockTCPProtocol` 或者 `sockUDPProtocol`。也可以在程序代码中设置 Protocol 属性, 例如, `Winsock1.Protocol = sockTCPProtocol` 将 Winsock1 控件设置为采用 TCP。

2. 采用 TCP 的程序设计

如果采用 TCP 设计程序, 应首先确定该程序是用于服务器端还是用于客户端。如果要创建一个服务器端程序, 那么该程序就需要“侦听”指定的端口。当客户端程序提出连接

请求时,该程序能够接受请求并建立连接。在连接建立之后,该程序与客户端程序就可以自由地相互通信了。

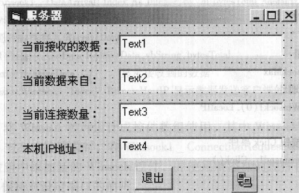
【例 9-1】 建立一个使用 Winsock 控件并采用 TCP 进行简单通信的程序实例,包括一个服务器端程序和一个客户端程序。

本例中服务器端程序可以接受来自多个客户端程序的连接请求和数据,并将接收到的数据内容显示在文本框中,同时可以显示数据来自于哪个客户端、当前服务器端建立了多少连接以及服务器端自身的 IP 地址。

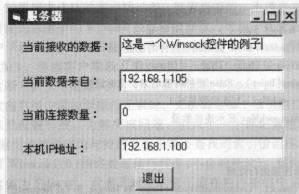
本例中,客户端程序可以指定要连接的服务器端的 IP 地址,对服务器发起连接请求,连接成功后,可以向服务器端发送数据。

(1) 服务器端程序设计

服务器端程序设计界面如图 9-1a 所示,在窗体左侧由上至下依次建立 4 个标签控件 Label1 ~ Label4,右侧由上至下依次建立 4 个文本框控件 Text1 ~ Text4,下面中间位置建立一个命令按钮。此外,还要添加一个运行时不可见控件 Winsock1,并在属性窗口将其 Index 属性设置为 0,以便在程序代码中用 Load 方法动态地完成控件添加。服务器端程序运行界面如图 9-1b 所示。



a)



b)

图 9-1 服务器端程序界面

a) 设计界面 b) 运行界面

服务器端程序代码如下:

```

Option Explicit
'变量 sockMax 对连接数量计数
Dim sockMax As Integer
'数组 IsIndexUsing 用于指示控件数
'组索引值是否可用
'IsIndexUsing(i) = False, 则 i 可用
'否则 i 不可用
Dim IsIndexUsing(1 To 15) _
As Boolean
Private Sub Form_Load()
'出错处理
On Error Resume Next
'初始化本地主机端口
Winsock1(0).LocalPort = 999
'开始侦听
Winsock1(0).Listen
Text1.Text = ""
Text2.Text = ""
'显示连接数量
Text3.Text = sockMax
'显示本机 IP 地址
Text4.Text = Winsock1(0).LocalIP
End Sub
'单击“退出”按钮 Click 事件
Private Sub Command1_Click()
End
End Sub
'对客户端断开连接进行处理
Private Sub Winsock1_Close(Index As Integer)
If Winsock1(Index).State < > sckClosed Then
Winsock1(Index).Close '关闭相应控件的连接
Unload Winsock1(Index) '卸载已关闭的控件
IsIndexUsing(Index) = False '将控件数组索引值标记为可用
sockMax = sockMax - 1 '连接数量减 1
Text3.Text = sockMax '显示连接数量
End If
End Sub
'对客户端请求连接进行处理
Private Sub Winsock1_ConnectionRequest(Index As Integer, ByVal RequestID As Long)
Dim i As Integer
If Index = 0 Then '只接收来自指定侦听端口的请求
If sockMax < 15 Then '由于资源限制, 设定最多可满足 15 个客户的连接请求

```

```

For i = 1 To 15
    If IsIndexUsing(i) = False Then '查找可使用的控件数组索引值
        Load Winsock1(i) '添加新的 Winsock 控件
        Winsock1(i).LocalPort = 0 '用端口 0 来动态地建立连接
        '用新添加的 Winsock 控件接受客户端的连接
        Winsock1(i).Accept RequestID
        IsIndexUsing(i) = True '将索引值设置为不可用
    Exit For
End If
Next i
sockMax = sockMax + 1 '连接数量加 1
Text3.Text = sockMax '显示连接数量
End If
End If
End Sub
'对接收到的数据进行处理
Private Sub Winsock1_DataArrival(Index As Integer, ByVal bytesTotal As Long)
    Dim ReceiveData As String
    '将接收到的数据存入 ReceiveData
    Winsock1(Index).GetData ReceiveData, vbString, bytesTotal
    Text1.Text = ReceiveData '显示接收到的数据
    Text2.Text = Winsock1(Index).RemoteHostIP '显示数据发送客户端的 IP 地址
End Sub

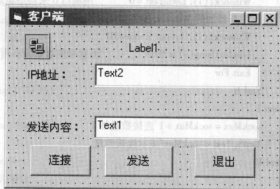
```

在上述程序中，将 Winsock 控件设计成控件数组使用，其中 Winsock1(0) 专门用于侦听客户端的连接请求。当有连接请求时，在 Winsock1_ConnectionRequest 事件过程中使用 Load 方法动态地向控件数组中添加新的 Winsock 控件，并使用新添加的控件接受客户端的连接，而 Winsock1(0) 始终保持对设定端口的侦听。

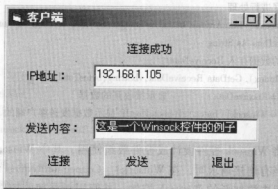
为了节省资源，控件数组被设定为最多只能添加 15 个新控件，即服务器端最多满足 15 个客户的连接请求。如果请求连接的客户端过多，则需要等待。不需要再进行数据传输的客户端断开连接后，对应的数组中的 Winsock 控件将被释放，可用于接受新的连接请求。在程序中，为统计数组中 Winsock 控件的数量设计了一个计数器 sockMax。每当客户端连接请求成功时，sockMax 就加 1，每当客户端断开连接时，sockMax 就减 1，以指示连接资源的数量。由于客户端连接断开的次序是随机的，所以被释放的 Winsock 控件在数组中的索引值也是不确定的，当再次向数组中添加新控件元素时，无法确定哪个索引值当前可用，哪个索引值对应的控件元素依然没被释放。为解决该问题，引入了一个标记数组 IsIndexUsing，该数组共 15 个元素，其索引值被设定为与控件数组中需要查找的索引值范围相同，即 1~15（控件数组中第一个元素 Winsock1(0) 始终处理侦听任务，不会被释放）。该标记数组类型为布尔型，当某一数组元素值为 False 时，代表其对应的索引值在控件数组中是可用的索引值，新添加的控件就可以使用该索引值作为下标。当标记数组中某一数组元素值为 True 时，代表其对应的索引值在控件数组中正被使用，不可以再用于新控件的添加。

(2) 客户端程序设计

客户端程序设计界面如图 9-2a 所示, 在窗体中建立 3 个标签控件 Label1 ~ Label3, 两个文本框 Text1 和 Text2, 由左至右 3 个命令按钮 Command1 ~ Command3。此外, 也要添加一个运行时不可见控件 Winsock1。客户端程序运行界面如图 9-2b 所示。



a)



b)

图 9-2 客户端程序界面

a) 设计界面 b) 运行界面

客户端程序相对简单一些, 只需根据用户输入的服务器 IP 地址和约定的端口号用 Connect 方法完成对服务器的连接, 就可以用 SendData 方法向服务器端发送数据信息了。程序中用 Label1 控件指示网络状态, 如当客户端程序没有连接服务器时, Label1 控件显示“尚未连接”, 而建立连接后显示“连接成功”。

客户端程序代码如下:

```
Dim RemoteHost As String
```

```
Dim RemotePort As String
```

```
Private Sub Form_Load()
```

```
Text1.Text = "
```

```
Text2.Text = "
```

```
RemotePort = "999"
```

```
Label1.Caption = "尚未连接"
```

```

End Sub
Private Sub Winsock1_Close()
    If Winsock1.State < > Closed Then
        Winsock1.Close
        Command2.Enabled = False
        '关闭提示
        Label1.Caption = "连接关闭"
    End If
End Sub
'结束提示
Private Sub Winsock1_SendComplete()
    Label1.Caption = "发送结束"
End Sub
'退出按钮
Private Sub Command1_Click()
    RemoteHost = Trim(Text2.Text)
    If Winsock1.State < > sckConnected Then
        If Winsock1.State < > sckClosed Then
            Winsock1.Close
        End If
        Winsock1.Connect RemoteHost, RemotePort
        Command2.Enabled = True
        Label1.Caption = "连接成功"
    Else
        Label1.Caption = "已连接"
    End If
End Sub
Private Sub Command2_Click()
    Winsock1.SendData Text1.Text
End Sub
Private Sub Command3_Click()
    End
End Sub

```

3. 采用 UDP 的程序设计

创建 UDP 应用程序比创建 TCP 应用程序要简单，因为 UDP 不需要显式的连接。

在上面的 TCP 应用程序中，服务器端必须有一个 Winsock 控件显式地进行“侦听”，而客户端必须使用 Connect 方法初始化连接。

UDP 不需要显式的连接。要在两个控件中间发送数据，需要连接的双方完成以下 3 步：

- 1) 将 RemoteHost 属性设置为对方计算机的名称。
- 2) 将本机的 RemotePort 属性设置为对方的 LocalPort 属性。
- 3) 调用 Bind 方法，指定使用的 LocalPort。

在创建 UDP 应用程序时调用 Bind 方法，这是必须的。Bind 方法的作用是为控件“保

留”一个本地端口。例如，如果将控件绑定到 1001 号端口，那么其他应用程序将不能使用该端口进行“侦听”。该方法阻止其他应用程序使用同样的端口。

在使用 UDP 的时候，可以任意地改变 RemoteHost 和 RemotePort 属性，同时始终保持绑定在同一个 LocalPort 上。TCP 与此不同，在改变 RemoteHost 和 RemotePort 属性之前，必须先关闭连接。

采用 UDP 进行通信的两台计算机，其地位可以看成是“平等的”，因此这种应用程序也被称为点到点的通信程序。

【例 9-2】 建立一个使用 Winsock 控件并采用 UDP 进行简单通信的程序实例。该例子包括通信双方两个程序，假定称为 A 和 B。

A、B 两程序均如下设计：

在窗体中放入一个 Winsock 控件，并将其协议属性设置为 UDPProtocol；在窗体中添加两个文本框控件，分别命名为 txtSend 和 txtOutput；为了加以区别，将两个程序的窗体标题分别设置为“A”和“B”。界面设计完毕，为窗体增加如下的代码：

```
Private Sub Form_Load()  
    '控件的名字为 Winsock1  
    With Winsock1  
        '重点：必须将 RemoteHost 的值修改为实际使用的计算机的名字  
        .RemoteHost = "202.113.72.9"  
        .RemotePort = 1002 '连接的端口号  
        .Bind 1001 '绑定到本地的端口  
    End With  
End Sub  
  
Private Sub txtSend_Change()  
    '在键入文本时，立即将其发送出去  
    Winsock1.SendData txtSend.Text  
End Sub  
  
Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)  
    Dim strData As String  
    Winsock1.GetData strData  
    txtOutput.Text = strData  
End Sub
```

以上代码中，程序 A 和程序 B 只在 RemoteHost、RemotePort 属性设置和调用 Bind 方法所用的参数这 3 个地方不同。

对于 RemoteHost 属性，程序 A 和程序 B 要互设为对方的计算机名或 IP 地址。假设运行程序 A 的计算机其 IP 地址为 192.168.1.101，运行程序 B 的计算机其 IP 地址为 192.168.1.102。则在程序 A 中，RemoteHost = "192.168.1.102"；在程序 B 中，RemoteHost = "192.168.1.101"。如果要在没有联网的计算机上同时运行调试这两个程序，可将 RemoteHost 属性均设置为 "127.0.0.1"

对于 RemotePort 属性，程序 A 和程序 B 要互设为对方要使用的端口号。即程序 A 的 RemotePort 属性值应和程序 B 中 Bind 方法的参数值相同；程序 B 的 RemotePort 属性值应和程

程序 A 中 Bind 方法的参数值相同。例如, 在程序 A 中 RemotePort = 1002, 则在程序 B 中应 Bind 1002, 反之亦然。

9.3 多媒体控件

VB6.0 提供了两个能操作多媒体文件的控件: Animation 控件和 Multimedia MCI 控件。利用这两个控件, 可以进行多媒体程序设计。

以上两个控件都是 ActiveX 控件而非 VB 内部控件, 使用它们之前必须确保在工程中添加了它们。方法是执行“工程”菜单的“部件”命令, 打开“部件对话框”选择 Microsoft Windows Common Controls-2 6.0 (即 Animation 控件)、Microsoft Multimedia Control 6.0 (即 Multimedia MCI 控件)。

9.3.1 Animation 控件

Animation 控件主要用于显示动画, 该控件只能播放无声的 AVI 文件。

AVI 文件是 Microsoft 支持的一种视频文件格式, 用于保存动画、电影片段等内容。AVI 动画类似于电影, 由若干帧位图组成。在 Windows 95 及以后版本的 Windows 系统中复制文件时, 可以看到该控件的一个例子: 在两个文件夹之间有一张纸(“文件”)在“飘动”。

在运行时, Animation 控件不具有自己的图文框。在播放时, Animation 控件使用了一个独立的线程。因此, 应用程序不会被阻塞, 可以继续在自己的进程中运行。

该控件的用途包括: 在对话框中显示出操作的长短和特征; 播放有关应用程序的无声动画, 提供使用指导; 使用户能够播放放入该控件的文件。

1. Animation 控件的属性、方法和事件

Animation 控件的属性、事件和方法较少, 也很简单。常用的属性、方法如下:

(1) AutoPlay 属性

功能: 在将“.avi”文件加载到控件时, 返回或设置一值, 该值确定 Animation 控件是否立即开始播放“.avi”文件。

数据类型: Boolean, 说明是否已激活 Autoplay 属性。

说明:

1) 该属性为 True 表示一旦“.avi”文件用 Open 方法打开, 就一直连续循环自动播放; 该属性为 False 表示用 Open 方法打开一个“.avi”文件后, 直到使用 Play 方法才能播放。

2) 用 Autoplay 属性播放的“.avi”文件将不断重复, 直到 Autoplay 的设置为 False 时为止。

(2) BackStyle 属性

功能: 返回或设置一个值, 该值确定 Animation 控件是在透明的背景上还是在动画剪辑中所指定的背景颜色上绘制动画。在运行时为只读。

数据类型: Integer (Boolean)。

说明:



1) 该属性默认值为 0, 表示背景是透明的, 即控件的背景颜色是可见的; 该属性值为 1 表示背景是不透明的, 即动画剪辑中指定的背景颜色将充满控件并隐没其背后的所有颜色。

2) 可以用 BackStyle 属性运行动画, 该动画显示 Animation 控件的背景颜色而非动画自身的背景颜色。

(3) Open 方法

格式如下:

对象名 . Open file

功能: 打开一个要播放的 “.avi” 文件。

说明:

1) 对象名: 即 Animation 控件对象的名称(下同)。

2) file: 该参数是必需的, 用于指示要播放的 “.avi” 文件的文件名。

3) 如果 AutoPlay 属性设置为 True, 则只要用 Open 方法打开该文件就开始播放它。在关闭 “.avi” 文件或设置 Autoplay 属性为 False 之前, 它都将不断重复播放。

4) Animation 控件不能播放含有声音数据的 “.avi” 文件。此外, 动画控件只能显示未压缩的或用行程编码(RLE)压缩的 “.avi” 文件。当用文件调用 Open 方法时, 如果该文件含有声音数据或不具有受支持的压缩格式, 则返回错误。在 Visual Basic CD-ROM 的 \ Graphics \ AVI 目录中可以找到许多该控件支持的无声 “.avi” 文件。要播放所有的 “.avi” 文件, 可以使用 Multimedia MCI 控件。

(4) Play 方法

格式如下:

对象名 . Play [= Repeat, Start, End]

功能: 在 Animation 控件中播放 “.avi” 文件。

说明:

1) Repeat: 可选项, 整型参数, 用于指定重复剪辑的次数。默认值是 -1, 使重复剪辑次数不受限定。

2) Start: 可选项, 整型参数, 用于指定开始的帧。默认值是 0, 表示在第一帧上开始剪辑。最大值是 65535。

3) End: 可选项, 整型参数, 用于指定结束的帧。默认值是 -1, 表示上一次剪辑的帧。最大值是 65535。

4) 用 Stop 方法终止播放 “.avi” 文件。

(5) Stop 方法

格式如下:

对象名 . Stop

功能: 在 Animation 控件中终止播放 “.avi” 文件。

说明: Stop 方法仅终止那些用 Play 方法启动的动画。当设置 Autoplay 属性为 True 时, 任何使用 Stop 方法的尝试都导致返回错误(35759)。

(6) Close 方法

格式如下:

对象名 . Close

功能: Close 方法使 Animation 控件关闭当前打开的“.avi”文件。如果没有加载任何文件,则 Close 不执行任何操作,也不会产生任何错误。

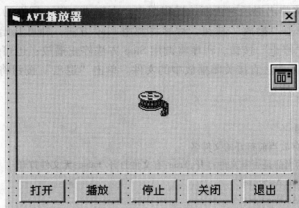
说明:为了终止播放文件,要用 Stop 方法。但是,如果 Autoplay 属性的设置为 True,则将其设置为 False 来停播文件。

2. Animation 控件的使用

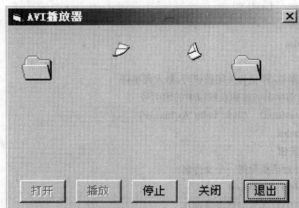
在使用该控件时,可用 Open 方法打开“.avi”文件,用 Play 方法进行播放,用 Stop 方法停止播放。在动画播放完毕以后,可用 Close 方法关闭该文件。在打开新文件之前不必关闭旧文件。

【例9-3】建立一个使用 Animation 控件播放“.avi”文件的程序。

AVI 播放器程序设计界面如图 9-3a 所示。在窗体中建立 5 个按钮 Command1(0)~Command1(3)及 Command2,一个 Animation 控件 Animation1,一个通用对话框控件 CommonDialog1。各个控件的属性设置见表 9-5。



a)



b)

图 9-3 AVI 播放器界面

a) 设计界面 b) 运行界面



表 9-5 例 9-3 各控件的属性设置

控件名称	属性名	属性值	控件名称	属性名	属性值
Form1	Caption	AVI 播放器	Command1 (1)	Enabled	False
	BorderStyle	1—Fixed Single	Command1 (2)	Caption	停止
CommonDialog1	Filter	AVI 文件 *.avi		Enabled	False
Animation1	AutoPlay	False	Command1 (3)	Caption	关闭
Command1 (0)	Caption	打开		Enabled	False
	Enabled	True	Command2	Caption	退出
Command1 (1)	Caption	播放		Enabled	True

程序运行时,单击“打开”按钮,在弹出的打开文件对话框中选中要播放的“.avi”文件,程序将调用 Open 方法打开要播放的文件。如果用户选择了 Animation 控件不支持的“.avi”文件,程序将弹出提示对话框,提醒用户。由于 AutoPlay 属性为 False,故打开的文件不会自动播放。单击“播放”按钮,程序将调用 Play 方法,即可开始播放。文件在播放过程中,可以单击“停止”按钮,程序将调用 Stop 方法停止播放;也可以单击“关闭”按钮,程序将调用 Close 方法直接关闭播放中的文件。单击“退出”按钮将结束程序运行。程序运行界面如图 9-3b 所示。

程序代码如下:

```

'定义两个变量
'strFileName 存放当前打开的文件名
'blsOpen 表示当前是否有文件打开,True:有文件打开/False:无文件打开
Dim strFileName As String
Dim blsOpen As Boolean
'在装载窗体时,给 blsOpen 变量置初值
Private Sub Form_Load()
    '没有文件打开
    blsOpen = False
End Sub
'当单击打开、播放、停止和关闭按钮时,触发此事件
'Index 指明当前单击的按钮在数组中的索引
Private Sub Command1_Click(Index As Integer)
    Select Case Index
        Case 0 '打开按钮
            '用打开文件对话框取得一个文件名
            CommonDialog1.ShowOpen
            '文件名为空吗? 如为空则放弃操作
            If CommonDialog1.FileName < > "" Then
                '把文件名存放在 strFileName 中
                strFileName = CommonDialog1.FileName
            '捕获错误信息

```

On Error GoTo errMes

'如果当前已经有文件打开,则首先用 Close 方法关闭它

If blsOpens Then

 Animation1. Close

End If

'用 Open 方法打开 AVI 文件

Animation1. Open strFileName

'设置打开状态为 True

blsOpens = True

'打开文件成功后,可以进行播放,使播放按钮变成可用

Command1 (1). Enabled = True

End If

Case 1 '播放按钮

'用 Play 方法播放 AVI 文件

Animation1. Play

'在播放时,使停止、关闭按钮可用,播放、打开按钮不可用

Command1 (0). Enabled = False

Command1 (1). Enabled = False

Command1 (2). Enabled = True

Command1 (3). Enabled = True

Case 2 '停止按钮

'用 Stop 方法停止播放

Animation1. Stop

'在停止时,使停止、关闭按钮不可用,播放、打开按钮可用

Command1 (0). Enabled = True

Command1 (1). Enabled = True

Command1 (2). Enabled = False

Command1 (3). Enabled = False

Case 3 '用 Close 方法关闭“.avi”文件

Animation1. Close

'恢复初始状态

Command1 (0). Enabled = True

Command1 (1). Enabled = False

Command1 (2). Enabled = False

Command1 (3). Enabled = False

'打开文件标志为 False

blsOpens = False

Case others

End Select

Exit Sub

'出错处理

errMes:

MsgBox("文件格式不支持或无此文件!")

```

End Sub
'退出应用程序
Private Sub Command2_Click()
    If blsOpens = True Then
        '调用 Close 方法关闭 Animation1
        Animation1.Close
    End If
    Unload Me
End Sub

```

9.3.2 Multimedia MCI 控件

Animation 控件的功能比较简单, 要在计算机上实现真正的多媒体效果, 需要使用 Multimedia MCI 控件。

Multimedia MCI 控件管理媒体控制接口 (Media Control Interface) 设备上的多媒体文件的记录与回放。从概念上说, 这种控件就是一组按钮, 它被用来向诸如声卡、MIDI 序列发生器、CD-ROM 驱动器、视频 CD 播放器和视频磁带记录器及播放器等设备发出 MCI 命令。MCI 控件还支持 Windows (*.avi) 视频文件的回放。

利用 Multimedia MCI 控件, 可以打开以下类型的设备或文件: AVI 视频文件、音频 CD 播放器、数字音频磁带播放器、窗口中的数字视频、覆盖设备、图像扫描仪、音响设备数字接口 (MIDI) 序列发生器、视频磁带录放器、视盘播放器或播放数字波形文件的音频设备等。

设计时, 将 Multimedia MCI 控件添加到窗体上, 其外观如图 9-4 所示。



图 9-4 Multimedia MCI 控件外观

控件中的按钮被分别定义为 Prev、Next、Play、Pause、Back、Step、Stop、Record 和 Eject。

1. Multimedia MCI 控件的属性、方法和事件

(1) AutoEnable 属性

功能: 这一属性决定 Multimedia MCI 控件是否能够自动启动或关闭控件中的某个按钮。如果 AutoEnable 属性被设置为 True, Multimedia MCI 控件就启用指定 MCI 设备类型在当前模式下所支持的全部按钮。这一属性还会禁用那些 MCI 设备类型在当前模式下不支持的按钮。

数据类型: Integer (Boolean)。

说明: 该属性为 True 将自动启用功能可用的按钮, 禁用功能不可用的按钮; 该属性为 False 将不能启用或禁用按钮。

(2) ButtonEnabled 属性

功能: 决定是否启用或禁用控件中的某个按钮 (禁用的按钮以淡化的形式显示)。

数据类型: Integer (Boolean)。

说明:

1) 该属性为 False (默认值) 将禁用 Button 所指定的按钮, 这个按钮的功能在控件中是不可用的; 为 True 将启用 Button 所指定的按钮, 这个按钮的功能在控件中是可用的。禁用的按钮以淡化的形式显示。

2) 对于这种属性, Button 可以是以下任何一种: Back、Eject、Next、Pause、Play、

Prev、Record、Step 或 Stop。

例如, 为了禁用 Eject 按钮, 可以使用以下语句:

```
[form.]MMControl.EjectEnabled = False
```

为了检查是否启用了 Pause 按钮, 可以使用以下语句:

```
If [form.]MMControl.PauseEnabled Then...
```

3) ButtonEnabled 属性的作用可以被 Enabled 和 AutoEnable 属性的作用所替代。只有当启用了 Multimedia MCI 控件(Enabled 属性被设置为 True)并且关闭了 AutoEnable 属性(设置为 False)时, 某个 ButtonEnabled 属性才能够启用或禁用 Multimedia MCI 控件中相关的按钮。

(3) Enabled 属性

功能: 决定控件的各个按钮是否可使用。

数据类型: Integer(Boolean)。

说明:

1) 该属性为 False 则控件中的所有按钮均被禁用(淡化的形式显示); 为 True(默认值)控件被启用。使用 AutoEnable 属性可以让 Multimedia MCI 控件能够自动地启用或禁用控件中的按钮。或者使用 ButtonEnabled 属性来启用或禁用控件中的某个按钮。

2) 该属性与 AutoEnable 属性和 ButtonEnabled 属性的关系:

当 Enable 属性被设置为 False, 那么不管 AutoEnable 和 ButtonEnable 属性如何设置, Multimedia MCI 控件都不允许访问它的按钮。

当 Enable 属性被设置为 True 时, AutoEnable 和 ButtonEnable 属性的设置才有效。这时, 当 AutoEnable 属性为 True 时, 不管 ButtonEnable 属性如何设置, 这种设置都无效。只有当 AutoEnable 属性为 False 时, ButtonEnable 属性设置才有效。

(4) CanPlay、CanRecord、CanStep 和 CanEject 属性

功能: 分别判别是否可以播放、可以进行记录、每次可以显示一帧和将媒体弹出。

数据类型: Integer(Boolean), 只读。

说明: 在 AutoEnable 为 False 时, 可以通过判别这些属性来设置 ButtonEnable 属性。

(5) Command 属性

功能: 指定将要执行的 MCI 命令。在设计时, 该属性不可用。

数据类型: String。

说明: 要用一个字符串参数给该属性赋值, 此字符串参数给出了将要执行的 MCI 命令的名称, 可以是 Open、Close、Play、Pause、Stop、Back、Step、Prev、Next、Seek、Record、Eject、Sound 或 Save。这些命令被立即执行, 并将错误代码存放在 Error 属性中。如要播放当前打开的设备, 可以用以下的语句:

```
MMControl1.Command = "Play"
```

(6) DeviceID 属性

功能: 指定当前打开的 MCI 设备的设备 ID。在设计时, 该属性不可用, 在运行时, 它是只读的。

数据类型: Integer。

说明: 如果没有打开任何设备, 则该参数为 0。

(7) DeviceType 属性

功能：指定要打开的 MCI 设备的类型。

数据类型：String。

说明：

1) 要用一个字符串参数给该属性赋值，此字符串参数给出了将要打开的 MCI 设备的类型，可以是 cdaudio、dat、DigitalVideo、Overlay、Scanner、Sequencer、VCR、AVIVideo、Videodisc、Waveaudio、Other 这些字符串。

2) 打开简单设备(如不使用文件的音频 CD)时，该属性必须设置。如果文件的扩展名没有指定将要使用的设备，那么打开复杂 MCI 设备时也必须设置该属性。

(8) Frames 属性

功能：规定 Step 命令能够前向单步或 Back 命令能够后向单步的帧数。在设计时，该属性不可用。

数据类型：Long。

(9) FileName 属性

功能：指定 Open 命令将要打开的或者 Save 命令将要保存的文件。如果在运行时要改变 FileName 属性，就必须先关闭然后再重新打开 Multimedia MCI 控件。

数据类型：String。

(10) From、To、Length 和 Position 属性

功能：如 Multimedia MCI 控件的 TimeFormat 属性定义一样，为 Play 或 Record 命令规定起始点、终点、长度和当前位置。在设计时，该属性不可用。

数据类型：Long。

(11) TimeFormat 属性

功能：规定用来报告所有位置信息的时间格式。

数据类型：Long(Enumerated)。

说明：TimeFormat 属性的设置值见表 9-6。

表 9-6 TimeFormat 属性的设置值

值	设置值	时间格式
0	mciFormatMilliseconds	毫秒数用 4 字节整数变量保存
1	mciFormatHms	小时数、分钟数和秒数被压缩到一个 4 字节整数中。从最低有效字节到最高有效字节，这 4 个数分别是：小时数(最低有效字节)；分钟数；秒数；未使用(最高有效字节)
2	mciFormatMsf	分钟数、秒数和帧被压缩到一个 4 位的整数中。从最低有效字节到最高有效字节，这 4 个数分别是：分钟数(最低有效字节)；秒数；帧；未使用(最高有效字节)
3	mciFormatFrames	帧用 4 字节的整数变量保存
4	mciFormatSmppte24	24-帧 SMPTE 将以下数值压缩到一个 4 字节的整数中。从最低有效字节到最高有效字节，这 4 个数分别是：小时数(最低有效字节)；分钟数；秒数；帧(最高有效字节)。SMPTE(动画和电视工程师协会)时间是一种绝对的时间格式，它按小时数、分钟数、秒数和帧的格式显示。标准的 SMPTE 的分度类型有 24、25 和 30 帧每秒

(续)

值	设置值	时间格式
5	mciFormatSmpte25	25-帧 SMPTE 按照与 24-帧 SMPTE 相同的顺序将数据压缩到一个 4 字节变量中
6	mciFormatSmpte30	30-帧 SMPTE 按照与 24-帧 SMPTE 相同的顺序将数据压缩到一个 4 字节变量中
7	mciFormatSmpte30Drop	30-放下-帧 SMPTE 按照与 24-帧 SMPTE 相同的顺序将数据压缩到一个 4 字节变量中
8	mciFormatBytes	字节数用 4 字节整数变量保存
9	mciFormatSamples	示例用 4 字节整数变量保存
10	mciFormatTmsf	曲目、分钟数、秒数和帧被压缩到一个 4 字节整数中。从最低有效字节到最高有效字节, 它们分别是: 曲目(最低有效字节); 分钟数; 秒数; 帧(最高有效字节)

(12) Mode 属性

功能: 返回打开的 MCI 设备的当前模式。在设计时, 该属性不可用, 在运行时, 它是只读的。

数据类型: Long。

说明: Mode 属性的值见表 9-7。

表 9-7 Mode 属性的值

值	设置值/设备模式	说 明	值	设置值/设备模式	说 明
524	mciModeNotOpen	设备没有打开	528	mciModeSeek	设备正在搜索
525	mciModeStop	设备停止	529	mciModePause	设备暂停
526	mciModePlay	设备正在播放	530	mciModeReady	设备准备好
527	mciModeRecord	设备正在记录			

(13) Notify 属性

功能: 决定下一条 MCI 命令是否使用 MCI 通知服务。如果它被设置为 True, 那么 Notify 属性在下一条 MCI 命令完成时产生一个回调事件(Done)。在设计时, 该属性不可用。

数据类型: Integer(Boolean)。

说明:

1) 该属性为 False(默认值)则下一条命令不产生 Done 事件; 为 True 则下一条命令产生 Done 事件。

2) 赋给该属性的值只对下一条 MCI 命令有效。后面的 MCI 命令会一直忽略 Notify 属性, 除非赋给它另外一个值(不同的或可标识的)。

(14) Silent 属性

功能: 决定是否播放声音。

数据类型: Integer(Boolean)。

说明: 该属性为 False 将播放任何存在的声音; 为 True 声音被关闭。

(15) UpdateInterval 属性

功能：规定两次连续的 StatusUpdate 事件之间的毫秒数。

数据类型：Integer。

说明：赋给该属性的值规定事件之间的毫秒数。如果毫秒数是 0，就表明没有 StatusUpdate 事件发生。

(16) StatusUpdate 事件

格式如下：

对象名_ StatusUpdate()

出现时机：按 UpdateInterval 属性所给定的时间间隔自动地发生。

说明：

- 1) 对象名：Multimedia MCI 控件对象的名称(下同)。
- 2) 这一事件允许应用程序更新显示，以通知用户当前 MCI 设备的状态。应用程序可以从 Position、Length 和 Mode 等属性中获得状态信息。

(17) Done 事件

格式如下：

对象名_ Done(NotifyCode As Integer)

出现时机：当 Notify 属性为 True 的 MCI 命令结束时发生。

说明：参数 NotifyCode 表示 MCI 命令是否成功。可以是 9-8 中的任意设置值。

表 9-8 参数 NotifyCode 的设置值

值	设置值	说明	值	设置值	说明
1	mciSuccessful	命令成功的执行	4	mciAborted	命令被用户中断
2	mciSuperseded	命令被其他命令所替代	8	mciFailure	命令失败

MCI 能在单个窗体中支持多个 Multimedia MCI 控件实例，这样就可以同时控制多台 MCI 设备，每台设备需要一个控件。

2. Multimedia MCI 控件的应用

下面的示例演示了打开一台使用兼容数据文件的 MCI 设备的过程。将这些代码放到 Form_ Load 过程，应用程序就可以使用 Multimedia MCI 控件来对文件 chimes. wav 进行播放、记录和倒带。在试运行这个示例之前，首先应创建一个包含 Multimedia MCI 控件的窗体。

程序代码如下：

```
Private Sub Form_Load()
```

```
    '设置属性以打开 MCI
```

```
    MMControl1.Notify = False
```

```
    MMControl1.Wait = True
```

```
    MMControl1.Shareable = False
```

```
    MMControl1.DeviceType = "WaveAudio"
```

```
    MMControl1.FileName = "c:\Windows\Media\chimes. wav"
```

```
    '打开 MCI WaveAudio 设备
```

```
MMControl1.Command = "Open"
```

```
End Sub
```

为了正确管理多媒体资源,在退出应用程序之前,应该关闭那些已经打开的 MCI 设备。将下面的语句放到 Form_ Unload 过程,那么在退出包含 Multimedia MCI 控件的窗体之前,就可以关闭那些已经打开的 MCI 设备。

```
Private Sub Form_ Unload( Cancel As Integer)
```

```
MMControl1.Command = "Close"
```

```
End Sub
```

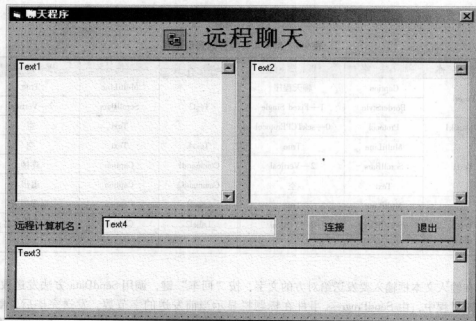
9.4 应用实例

【例 9-4】 使用 Winsock 控件,编写一个简易聊天程序。

简易聊天程序设计界面如图 9-5a 所示。在窗体上建立 4 个文本框 Text1 ~ Text4, 2 个命令按钮 Command1、Command2, 2 个标签控件 Label1、Label2, 一个 Winsock 控件 Winsock1。各个控件的属性设置见表 9-9。

界面设计完毕,下面就要对各个事件进行编程,以控制应用程序的运行。

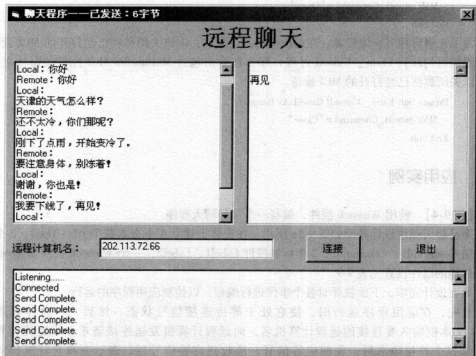
首先,在应用程序运行时,使它处于等待连接信号状态。然后,一台计算机在 Text4 文本框输入要连接的远程计算机名,向远程计算机发送连接请求。当远程计算机接收到这个连接请求后,发回应答信号。接收到应答信号后,表示在两台计算机间已经建立了连接。



a)

图 9-5 简易聊天程序界面

a) 设计界面



b)

图 9-5 简易聊天程序界面(续)

b) 运行界面

表 9-9 例 9-4 各控件的属性设置

控件名称	属性名	属性值	控件名称	属性名	属性值
Form1	Caption	聊天程序	Text3	MultiLine	True
	BorderStyle	1—Fixed Single		ScrollBars	2—Vertical
Winsock1	Protocol	0—sckTCPProtocol		Text	空
Text1	MultiLine	True	Text4	Text	空
	ScrollBars	2—Vertical	Command1	Caption	连接
	Text	空	Command2	Caption	退出
Text2	MultiLine	True	Label1	Caption	远程聊天
	ScrollBars	2—Vertical	Label2	Caption	远程计算机名:
	Text	空			

在输入文本框输入要发送给对方的文字，按“回车”键，调用 SendData 方法发送数据，在发送过程中，由 SendProgress 事件在标题栏显示当前发送的字节数。发送完毕后，触发 SendComplete 事件，显示完成发送信息；对方在接收到数据时，触发 DataArrival 事件，调用 GetData 方法接收数据，并在谈话内容文本框显示。当任意一方断开连接后，另一方会触发 Close 事件，负责关闭当前连接，等待下次连接信号。

程序中在 Error 事件中实时捕获错误信息,并在状态文本框加以显示。

程序代码如下:

```
Private Sub Form_Load()
    '设置本机的端口
    Winsock1.LocalPort = 1001
    '开始侦听是否有远程计算机连接到本机
    Winsock1.Listen
    '在状态文本框显示当前状态
    Text3.Text = Text3.Text & vbCrLf & "Listening....."
End Sub

'输入一个远程计算机的主机名后,按“回车”键进行连接
Private Sub Text4_KeyPress(KeyAscii As Integer)
    If KeyAscii = Asc(vbCr) Then
        '关闭侦听端口
        Winsock1.Close
        '设置远程计算机的主机名和端口
        Winsock1.RemoteHost = Text4.Text
        Winsock1.RemotePort = 1001
        '向远程计算机发送连接信号
        Winsock1.Connect
    End If
End Sub

'用鼠标单击“连接”按钮等同于在计算机名文本框输入远程计算机名后按“回车”键
Private Sub Command1_Click()
    If Text4.Text <> "" Then
        '向 Text4 文本框发送一个“回车”键
        Text4_KeyPress Asc(vbCr)
    End If
End Sub

'接收到远程计算机发送来的连接信号,发送应答信号
Private Sub Winsock1_ConnectionRequest(ByVal RequestID As Long)
    Winsock1.Close
    Text3.Text = Text3.Text & vbCrLf & "Connection Requested."
    '调用 Accept 方法回送应答信号
    Winsock1.Accept RequestID
End Sub

'接到应答信号后,触发此事件,并显示成功连接的信息
Private Sub Winsock1_Connect()
    Text3.Text = Text3.Text & vbCrLf & "Connected"
End Sub

'向远程计算机发送文字信息
Private Sub Text2_KeyPress(KeyAscii As Integer)
```

```

If KeyAscii = Asc( vbCr) Then
    '调用 SendData 方法把输入的文字发送给对方
    Winsock1. SendData Text2. Text
    '在谈话内容文本框显示发送到远程计算机的文本
    Text1. Text = Text1. Text & vbCrLf & "Local:" & Text2. Text
    '清空输入文本框的内容
    Text2. Text = ""
End If
End Sub

'当对方计算机发送来信息时,触发此事件,把接收到的文字显示在谈话内容文本框
Private Sub Winsock1_DataArrival( ByVal bytesTotal As Long)
    Dim strTmp As String
    '调用 GetData 方法接收数据
    Winsock1. GetData strTmp
    Text1. Text = Text1. Text & vbCrLf & "Remote:" & strTmp
End Sub

'发送信息完成后,在状态文本框显示完成信息
Private Sub Winsock1_SendComplete()
    Text3. Text = Text3. Text & vbCrLf & "Send Complete. "
End Sub

'在窗体的标题栏上实时显示当前发送的数据量
Private Sub Winsock1_SendProgress( ByVal bytesSent As Long, ByVal bytesRemaining As Long)
    Form1. Caption = "聊天程序——已发送:" & bytesSent & "字节"
End Sub

'当对方断开连接后,触发此事件,在这里关闭本次连接,开始侦听信号,等待下次连接
Private Sub Winsock1_Close()
    Winsock1. Close
    Text3. Text = Text3. Text & vbCrLf & "Close. "
    Winsock1. Listen
    Text3. Text = Text3. Text & vbCrLf & "Listen....."
End Sub

'出错处理
Private Sub Winsock1_Error( ByVal Number As Integer, _
    Description As String, ByVal Scode As Long, _
    ByVal Source As String, ByVal HelpFile As String, _
    ByVal HelpContext As Long, CancelDisplay As Boolean)
    '显示错误信息
    Text3. Text = Text3. Text & vbCrLf & Number & Description
End Sub

'退出应用程序
Private Sub Command2_Click ()
    End
End Sub

```

【例 9-5】 利用 Multimedia MCI 控件实现一个简易媒体播放器。

简易媒体播放器程序设计界面如图 9-6a 所示。在窗体中建立 3 个按钮 Command1 ~ Command3, 一个 Multimedia MCI 控件 MMControl1, 一个 Slider 控件 Slider1, 一个通用对话框控件 CommonDialog1。各个控件的属性设置见表 9-10。



a)



b)

图 9-6 简易媒体播放器界面

a) 设计界面 b) 运行界面

表 9-10 例 9-5 各控件的属性设置

控件名称	属性名	属性值	控件名称	属性名	属性值
Form1	Caption	简易媒体播放器	Slider1	Max	50
	BorderStyle	1—Fixed Single		Min	0
MMControl1	Frames	2	Command1	Caption	打开
	UpdateInterval	100	Command2	Caption	保存
Slider1	LargeChange	5	Command3	Caption	退出

程序运行时,单击“打开”按钮,在弹出的打开文件对话框中选中要播放的音频或视频文件,程序将用选中的文件名给 FileName 属性赋值,并通过给 Command 属性赋字符串 "Open",命令控件打开文件。在打开之前,程序会先进行判断,若已经打开了其他文件,则通过赋 "Close" 字符串给 Command 属性,关闭原来打开的文件后,再打开新文件。如果在此过程中发生错误,程序将给出提示。文件打开后,用户就可通过控件的相关操作按钮控制其播放或暂停。程序运行界面如图 9-6b 所示。

程序代码如下:

```
'每隔 100ms,触发此事件,根据当前 MCI 设备的位置,设置滚动条的游标
```

```
Private Sub MMControl1_StatusUpdate()
```

```
Slider1.Value = MMControl1.Position * Slider1.Max/MMControl1.Length
```


End Sub

'当单击“打开”菜单项时,触发此事件

Private Sub Command1_Click()

'调用 CommonDialog1 取得一个文件名

CommonDialog1.ShowOpen

'判别文件名是否为空

If CommonDialog1.FileName < > "" Then

'设置 mmMCI 要打开的文件名

MMControl1.FileName = CommonDialog1.FileName

'捕获错误信息

On Error GoTo errMCI

'如果当前已经有设备打开,则必须关闭后才能打开新设备

If MMControl1.DeviceID < > 0 Then

'发送 Close 命令关闭当前打开设备

MMControl1.Command = "Close"

End If

'打开设备

MMControl1.Command = "Open"

End If

Exit Sub

errMCI:

'出错处理

MsgBox (MMControl1.ErrorMessage)

End Sub

'关闭当前设备退出

Private Sub Command3_Click()

'关闭打开的设备

MMControl1.Command = "Close"

Unload Me

End Sub

'保存录音的文件

Private Sub Command2_Click()

If MMControl1.CanRecord Then

MsgBox "要保存文件吗?", vbOKCancel, "保存文件"

If vbOK Then

'发送 Save 命令保存文件

MMControl1.Command = "save"

End If

End If

End Sub

Private Sub Form_Unload(Cancel As Integer)

MMControl1.Command = "Close"

End Sub

习 题

- 9-1 改进例 9-4 中的简易聊天程序，使之可以同时和多人在线交谈。
- 9-2 编写一个程序，利用 Animation 控件循环播放一个动画文件。动画文件可由用户任意指定。
- 9-3 设计一个多媒体播放器，能够播放有声音的 AVI 文件，并可静音播放和重复播放。

第 10 章 数 据 库

数据库技术是数据处理的核心技术。数据库管理系统(DBMS, DataBase Management System)是帮助人们处理大量信息、实现管理现代化、科学化的强有力的工具。VB 是 Microsoft 公司的非常优秀的数据库开发平台。许多应用程序的开发者都选择 VB 作为开发数据库前台应用程序的工具。

本章在介绍数据库的一些基础知识之后,主要介绍几种在 VB 中应用的数据库访问技术:

- 1) 使用 VB 中的数据库访问控件进行数据库的访问。
- 2) 使用 DAO 技术对数据库访问。
- 3) 使用 ADO 技术对数据库访问。

10.1 数据库基础知识

进行 VB 数据库开发,首先要了解与之相关的数据库基础知识。

10.1.1 数据库基本概念

1. 数据库

所谓的数据库(DB, DataBase)就是存放在计算机的外存储器中的相关数据的集合。这里的数据是指描述事务的符号记录,它既包含平时所说的数字,还包括文字、影音、图像等形式。

2. 数据库管理系统

数据库管理系统(DBMS)是一个软件系统。它负责将收集并抽取的大量数据进行科学的组织,并将其存储在数据库中,高效地进行处理。它是数据库管理的核心,是为数据库建立、使用和维护的软件。它建立在操作系统的基础上,是位于操作系统和用户之间的数据库管理软件,负责对数据库进行统一的管理和控制。

DBMS 只提供对数据的管理功能,为了实现某种具体的功能,必须要有相应的数据库应用程序。

3. 数据库系统

数据库系统是指实际可运行的,按照数据库方式存储、维护和向应用程序提供数据或信息支持的计算机系统。一个完整的数据库系统有数据库、数据库管理系统、数据库应用程序、计算机软件和硬件系统及数据库管理员(DBA)组成。

4. 数据模型

数据库中的数据是按照一定的数据模型来组织的,数据模型是把现实世界转换为计算机能够处理的数据世界的桥梁。目前常用的数据模型有多种,分别是层次模型、网状模型、关系模型。现在用户使用最多的是关系模型。

5. 数据库应用程序

数据库应用程序是指用 VB、VC、FoxPro、Delphi 等开发工具设计的、用于实现某种特定功能的应用程序，如工资管理系统、进销存管理系统等。

10.1.2 关系数据库

目前大多数数据库管理系统都是基于关系模型的数据库。关系型数据库是根据表、记录和字段之间的关系进行数据组织和访问的一种数据库，它通过若干表来存储数据，并通过关系将这些表联系在一起。

一个关系数据库由若干个数据表组成，一个数据表又由若干个记录组成，而每一个记录又是由若干个以字段属性加以分类的数据项组成。

关系数据库涉及的主要概念如下：

- 1) 数据表。一个关系对应一个数据表，由一组相关的数据记录组成，每行有一个记录号，用以标识记录。
- 2) 记录。表中的每一行称为记录，它由若干个字段组成。
- 3) 字段。表中的每一列称为一个字段，其反映的是研究对象的某一方面的特性。
- 4) 主键。在表中能唯一地标识某一个记录的字段。
- 5) 外键。在表 A 中的某个字段，在该表中虽然不是主键，或是作为主键的一部分，但该字段在表 B 中是主键，那么这个字段在表 A 称为外键。
- 6) 索引。为了提高数据的访问效率，可以对数据表建立索引，从而改变表中记录的逻辑顺序。

图 10-1 所示为一个“学生信息”的关系结构。



图 10-1 “学生信息”的关系结构

关系数据库的主要功能如下：

- 1) 数据定义功能。数据定义语言可定义数据库中的数据对象。
- 2) 数据操纵功能。数据操纵语言可实现对数据库的数据查询、插入、删除和修改等操作。
- 3) 数据库的运行管理。保证数据的安全性、完整性，多用户对数据的并发使用，发生故障后的系统恢复。
- 4) 数据库的建立和维护功能。通过实用程序实现数据库数据批量装载、数据库转储、介质故障恢复、数据库的重组、性能监视等操作。



10.2 SQL 语言

SQL 是 Structured Query Language(结构化查询语言)的缩写。SQL 是一种标准的关系数据库语言,SQL 功能强大、简单易学、使用方便,已经成为数据库操作的基础,几乎所有的关系型数据库均支持 SQL。

在 VB 中利用 SQL 语言主要实现数据的查询、插入记录、删除记录、更新记录等操作。表 10-1 列出了常用的 SQL 的主要语句及说明。

表 10-1 SQL 的主要语句及说明

SQL 指令	说 明	SQL 指令	说 明
Select	查询数据,即从数据库中返回记录集	Delete	删除数据库中表的记录
Insert	向数据库的表中插入一条记录	Create	创建一个新的数据库
Update	修改数据库中表的记录	Drop	删除一个数据库

1. Select 查询语句

数据查询是数据库中最常见的操作,既可完成简单的单表查询,也可完成相当复杂的多表连接查询、嵌套查询和集合查询。

(1) Select 语句的基本格式

格式如下:

Select 字段名列表 From 数据表名

[Where 筛选条件] [Order by 字段名] [Asc | Desc]

说明:

- 1) Select 语句是由 Select 子句、From 子句和 Where 子句构成。
- 2) 字段名列表:用来描述查询获得的记录集的列,多列之间用逗号分割。可以用“*”号返回数据表中所有的列。
- 3) 筛选条件是指查询条件,只有满足此条件的记录才会返回到记录集中。
- 4) Order by 字段名:指返回的记录集按 Order by 中指出的字段名对记录集进行升序(ASC)或降序(DESC)排列。

(2) Select 语句应用举例

在图 10-1 中列写了“学生信息”表的结构。针对此图所示,则:

1) 检索表中所有学生的信息:

```
Select * From 学生信息
```

2) 检索表中专业为计算机应用的所有学生的学号、姓名和性别信息:

```
Select 学号,姓名,性别 From 学生信息 Where 专业 = '计算机应用'
```

2. Insert 插入语句

(1) 使用 Insert 语句可以向表中插入一条记录

该语句的语法格式如下:

Insert into 表名(字段名)

values(字段值)

说明:

- 1) 字段名: 指出向哪些字段插入值, 多个字段之间用逗号分割。
- 2) 字段值: 向表中插入一条记录的相应字段的值。
- 3) 插入的常量值的个数应和字段名的个数相同, 类型也应和字段名的类型一致。

(2) Insert 语句的应用举例

向图 10-1 所示的“学生信息”表中插入一条记录, 记录的信息如下:

学号: 050004; 姓名: 张新; 性别: 男; 年龄: 18; 专业: 计算机应用;

家庭住址: 天津市南开区。

对应的 Insert 语句如下:

```
Insert into 学生信息(学号,姓名,性别,年龄,专业,家庭住址)
```

```
values('050004','张新','男','18','计算机应用','天津市南开区')
```

3. Update 修改语句

在 SQL 语句中, 可以通过 Update 语句来修改表中满足条件的记录。

(1) Update 语句的格式如下:

```
Update <表名>
```

```
Set <列名>=<表达式>[, <列名>=<表达式>]...
```

```
[Where <条件>]
```

Update 语句完成的功能是修改指定表中满足 Where 子句条件的记录。其中, Set 子句给出了取代原来字段值的新的表达式的值。若省略 Where 子句, 则表示要修改所有的记录。

(2) Update 语句应用举例

- 1) 将图 10-1 中表“学生信息”所有学生的年龄都增 1。对应的 SQL 语句如下:

```
Update 学生信息
```

```
Set 年龄 = 年龄 + 1
```

- 2) 将图 10-1 中表“学生信息”中学号 = 050001 的学生的专业改为: 软件。对应的 SQL 语句如下:

```
Update 学生信息
```

```
Set 专业 = '软件' Where 学号 = '050001'
```

4. Delete 删除语句

数据删除也是一项常用的技术。比如学生信息发生变化, 如学生退学, 就要把相应的记录删除。数据删除在 SQL 中是用命令 Delete 来实现的。格式如下:

```
Delete From <表名>
```

```
[Where <条件>]
```

Delete 命令的功能是从指定的表中删除满足 Where 条件的所有记录。若省略 Where 语句, 表示删除表中的全部记录, 但表的定义仍然存在。

Delete 语句应用举例:

- 1) 从“学生信息”表中删除学号 = 050001 的学生记录。SQL 语句如下:

```
Delete From 学生信息
```

```
Where 学号 = '050001'
```

- 2) 从“学生信息”表中删除所有的记录。SQL 语句如下:



Delete From 学生信息

10.3 数据控件及应用

在 VB 中提供一些和数据库访问相关的控件，这些控件包括数据库连接控件和数据显示控件。主要包括 Data 控件、ADO 控件、DBGrid 控件、MSFlexGrid 控件等。

10.3.1 Data 控件

数据控件 Data 提供了一种方便访问数据库中数据的方法，使用数据库控件无需编写代码就可以对 VB 所支持的各种类型的数据库执行大部分数据访问操作。

数据控件本身不能显示和修改记录，只能在与数据库控件相关联的数据约束控件中显示各个记录。数据控件只相当一个记录指针，可以用鼠标操纵 Data 控件，由一条记录移动到另一条记录或移动到 RecordSet 的开始或结尾。

1. 工具箱和窗体上的 Data 控件

工具箱和窗体上的 Data 控件，分别如图 10-2 和图 10-3 所示。



图 10-2 工具箱中的 Data 控件

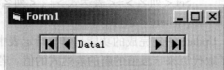


图 10-3 窗体上的 Data 控件

Data 控件图形中有 4 个按钮，其作用如下：

- 1) 将数据表中的记录指针移动到第一个记录。
- 2) 将数据表中的记录指针移动到当前记录的上一个记录。
- 3) 将数据表中的记录指针移动到当前记录的下一个记录。
- 4) 将数据表中的记录指针移动到最后一个记录。

2. Data 控件的属性

Data 控件和其他 VB 控件一样，也有一些属性。和访问数据库相关的属性包括：

(1) Connect 属性

用于指定数据库的类型，其值为一字符串，默认为 Microsoft Access 的 MDB 文件。

(2) DatabaseName 属性

用于设置或返回数据控件的数据源的名称及位置。

(3) RecordsetType 属性

用于返回或设置一个值，指出记录集的类型，这些记录集由数据控件创建。其中 0 为表 (Table) 类型，1 为动态集 (Dynaset) 类型，2 为快照 (Snapshot) 类型。

(4) RecordSource 属性

用来设置 Data 控件的数据来源，可以是数据库的表名或 SQL 语句。

3. Data 控件的常用方法

(1) Refresh 方法

当 DatabaseName 或 Connect 等属性的设置值发生改变时,可以在数据控件上使用 Refresh 方法来打开或重新打开数据库,以获得数据库中最新的数据。

(2) UpdateRecord 方法

当数据绑定控件的内容改变时,如果不移动记录指针,则数据库中的值不会改变,可通过调用 UpdateRecord 方法来确认对记录的修改,将约束控件中的数据强制写入数据库中。

(3) UpdateControls 方法

此方法用于从数据控件的 Recordset 对象中读取当前记录,并将数据显示在相关数据绑定控件上。

4. Data 控件的应用

利用 Data 控件可以直接或间接地访问数据库,如 Microsoft Access、SQL Server、Oracle 及 dBase、Excel 等。具体访问数据库情况说明如下:

1) Data 控件可以直接访问 Access、dBase、Excel 等数据库。此时,Data 属性设置如下:

Connect 属性:直接按属性值选择即可,如 Access。

DatabaseName 属性:设定数据库文件名,如 c:\db1.mdf。

Recordsource 属性:可以为对应的数据库的表,也可以为 Select 类型的 SQL 语句。

2) 对于 SQL Server、Oracle 等数据库,Data 控件不能直接访问,必须通过 ODBC(开放数据库链接)技术来访问。在使用 ODBC 技术之前必须先建立 ODBC 数据源,此时 Data 属性设置如下:

Connect 属性:

属性值的格式如下:

ODBC; DSN = ODBC 数据源名称; uid = 用户名; pwd = 密码

说明:ODBC 指出数据源的类型;DSN 指出了 ODBC 数据源的名称;uid 指出访问数据源所对应的数据库服务器的用户名;pwd 指出用户名对应的密码。

Recordsource 属性:可以为数据库中的表名,也可以为具体的 Select 类型的 SQL 语句。

10.3.2 ADO 控件

ADO 控件使用 Microsoft ActiveX 数据对象(ADO)来快速建立数据绑定的控件和数据提供者之间的连接。数据绑定控件是任何具有“数据源”属性的控件。数据提供者可以是任何符合 OLEDB 规范的数据源。ADO 控件比 Data 控件对数据的访问更灵活,功能更全面。

1. 工具箱和窗体上的 ADO 控件

ADO 控件不是基本的内部控件,工具箱中默认没有 ADO 控件,要想使用 ADO 控件必须将其加入到工具箱中。

依次选择“工程”→“部件”菜单选项,打开“部件”窗口,如图 10-4 所示。选择 Microsoft ADO Data Control 6.0(OLEDB)部件。

工具箱和窗体上的 ADO 控件,分别如图 10-5 和图 10-6 所示。

2. ADO 控件的属性

ADO 控件和 Data 控件一样,也有一些访问数据库的属性。和访问数据库相关的属性包括:

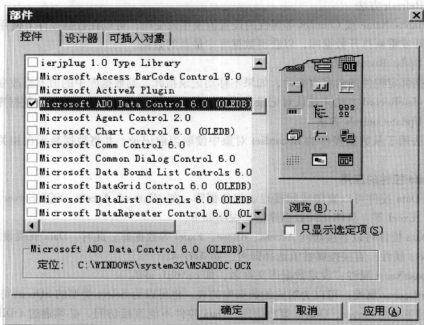


图 10-4 “部件”窗口

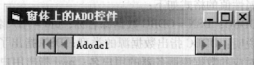


图 10-6 窗体上的 ADO 控件

图 10-5 工具箱中的 ADO 控件

(1) ConnectionString 属性

用于指定和数据库建立连接的连接串。在属性窗口中单击 ConnectionString 属性，显示“属性页”对话框，通过此对话框可以设置和数据库建立连接的连接串，如图 10-7 所示。

(2) RecordSource 属性

用来设置 ADO 控件的数据来源，可以是数据库的表名、SQL 语句或存储过程。在属性窗口中单击 RecordSource 属性，显示“属性页”对话框，如图 10-8 所示。

(3) ConnectionTimeout 属性

连接数据库超时的响应时间，单位为秒(s)。

3. ADO 控件的常用方法

Refresh 方法：

当 DatabaseName 或 Connect 等属性的设置值发生改变时，可以在数据控件上使用 Refresh 方法来打开或重新打开数据库，以获得数据库中最新的数据。



图 10-7 连接串属性对话框

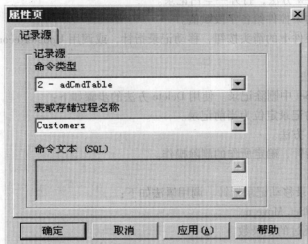


图 10-8 “属性页”对话框

10.3.3 数据控件的记录集 RecordSet 对象

VB 6.0 可以使用 RecordSet 对象来检索和显示数据库表中的记录。RecordSet 对象提供了一些属性和方法用来操作记录集。

1. 使用格式

使用 RecordSet 对象的属性和方法格式如下：

对象名.RecordSet. 属性或方法

2. RecordSet 对象的属性

(1) RecordCount 属性

用来返回 RecordSet 对象中的记录数目。无法确定记录数时该属性返回 -1。读取已关闭的 RecordSet 上的 RecordCount 属性将产生错误。

(2) Eof 属性和 Bof 属性

Eof 指示当前记录位置位于 RecordSet 对象的最后一个记录之后；Bof 指示当前记录位置位于 RecordSet 对象的第一个记录之前。

这两个属性经常用于判断记录指针是否越界。当 Bof 或 Eof 为真时，不能从结果集中读取数据，否则会产生错误。

(3) AbsolutePosition 属性

返回指定 RecordSet 对象当前记录的序号位置。

(4) Bookmark 属性

重新定位记录集的指针位置。

3. RecordSet 对象的方法

(1) AddNew 方法

该方法用来在 RecordSet 中插入新记录，前提是这个 RecordSet 对象是可更新的。在使用该方法后，用 Update 方法进行保存，新记录便成为当前记录。使用 AddNew 方法的步骤如下：

- 1) 调用 AddNew 方法，打开一空白记录。
- 2) 通过相关约束控件给各字段赋值。
- 3) 单击数据控件上的箭头按钮，移动记录指针，或调用 UpdateRecord 方法确定所作的修改。

(2) Delete 方法

用来从 RecordSet 中删除记录。使用 Delete 方法的步骤如下：

- 1) 将要删除的记录定位为当前记录。
- 2) 调用 Delete 方法。
- 3) 移动记录指针，确定所作的删除操作。

(3) Move 方法

Move 从当前记录移动记录指针。调用语法如下：

对象名.RecordSet.Move n

其中，n 是要跳过的记录数。

(4) MoveFirst 方法

MoveFirst 移动记录指针到第一条记录。

(5) MoveLast 方法

MoveLast 移动记录指针到最后一条记录。

(6) MovePrevious 方法

MovePrevious 移动记录指针到前一条记录。

(7) MoveNext 方法

MoveNext 移动记录指针到下一条记录。

(8) Edit 方法

用来编辑或修改 RecordSet 中的记录。使用 Edit 方法的步骤如下：

- 1) 将要编辑或修改的记录定位为当前记录。

- 2) 调用 Edit 方法。
- 3) 通过相关约束控件修改各字段的值。
- 4) 移动记录指针, 确定所作的修改操作。

(9) Close 方法

用于关闭指定的记录集并释放分配的资源。

10.3.4 数据绑定控件

数据绑定控件是指能够与数据库中的数据表的某个字段建立关联的控件。当这些数据绑定控件被绑定在数据控件上时, 数据控件能够将自身连接的数据源中的数据传送给这些数据绑定控件, 当数据控件的数据源中的数据改变时, 数据绑定控件中的数据也随之改变; 反之, 若数据绑定控件的值被修改, 这些修改后的数据也会自动地保存到数据库的数据表中。

在 VB 中, 通过操作数据绑定控件, 就可以操作对应数据库中的数据了。常用的数据绑定控件包括常用控件和 ActiveX 控件。

常用控件包括:

- TextBox 文本框控件;
- Label 标签控件;
- ListBox 列表框控件;
- ComboBox 组合框控件;
- CheckBox 复选框控件;
- PictureBox 图片框控件;
- Image 图像控件。

ActiveX 控件包括:

- DBGrid 数据库表格控件;
- DBCombo 数据库组合控件;
- DBList 数据库列表控件;
- DataGrid 数据表格控件;
- DataCombo 数据组合控件;
- DataList 数据列表控件。

常用控件通常通过 DataSource 和 DataField 两个属性和数据源中的某个字段建立绑定关系; 而 ActiveX 控件只通过 DataSource 属性和数据源建立绑定关系。

10.3.5 数据控件应用

通过数据控件(Data 控件或 ADO 控件)可以和数据库建立连接, 结合数据绑定控件就可以操作数据库了。

【例 10-1】 创建一个窗体, 利用数据控件 Data 创建一个学生信息管理的程序, 程序运行的结果如图 10-9 所示。

操作步骤:

- 1) 创建用来保存学生信息的 Access 数据库: DBStudent; 并创建表 Student 用来保存学生的基本信息; 表 Student 的结构见表 10-2。

图 10-9 学生信息管理程序运行结果

表 10-2 表 Student 的结构

字段名	字段类型	字段大小	是否为空	字段名	字段类型	字段大小	是否为空
学号	文本	6	否	系	文本	20	是
姓名	文本	10	否	住址	文本	50	是
性别	文本	2	否	电话	文本	12	是
E-Mail	文本	20	是	备注	文本	100	是

2) 创建窗体及控件, 并参照图 10-10 设置控件的相关属性。这里的每一个控件分别按其前面的标签控件的文本内容和表 Student 中的字段进行绑定。数据控件 Data 在运行时将其置为不可见。

3) 打开“代码设计”窗口, 输入程序代码。

Form_Load() 事件代码如下:

```
Private Sub Form_Load()
    Combo1.AddItem "男"
    Combo1.AddItem "女"
    Data1.DatabaseName = App.Path & "\DBStudent.mdb" '设置 Data 控件属性
    Data1.RecordSource = "Student"
End Sub
```

cmdAdd_Click() 事件代码如下:

```
Private Sub cmdAdd_Click() '“添加”按钮, 添加一条记录
    If cmdAdd.Caption = "添加" Then
        cmdAdd.Caption = "确定"
        Data1.RecordSet.AddNew
        Text1(0).SetFocus
    Else
```

图 10-10 学生信息管理设计窗体

```
cmdAdd.Caption = "添加"
```

```
Data1.RecordSet.Update
```

```
End If
```

```
End Sub
```

cmdEdit_Click()事件代码如下:

```
Private Sub cmdEdit_Click()
```

```
' "修改"按钮,修改当前记录
```

```
If MsgBox("是否真的修改?", vbQuestion + vbYesNo, "提示") = vbYes Then
```

```
Data1.RecordSet.Edit
```

```
Data1.RecordSet.MoveFirst
```

```
End If
```

```
End Sub
```

cmdDelete_Click()事件代码如下:

```
Private Sub cmdDelete_Click()
```

```
' "删除"按钮,删除当前记录
```

```
If MsgBox("是否真的删除?", vbQuestion + vbYesNo, "提示") = vbYes Then
```

```
Data1.RecordSet.Delete
```

```
Data1.RecordSet.MoveFirst
```

```
End If
```

```
End Sub
```

cmdFirst_Click()事件代码如下:

```
Private Sub cmdFirst_Click()
```

```
' "第一条"按钮,移到第一条
```

```
Data1.RecordSet.MoveFirst
```

```
cmdPrevious.Enabled = False
```

```
cmdNext.Enabled = True
```

```
End Sub
```

cmdPrevious_Click()事件代码如下:



```
Private Sub cmdPrevious_Click() '“上一条”按钮,移到上一条
```

```
    If Data1.RecordSet.BOF = True Then
```

```
        cmdPrevious.Enabled = False
```

```
        Data1.RecordSet.MoveFirst
```

```
    Else
```

```
        Data1.RecordSet.MovePrevious
```

```
    End If
```

```
    cmdNext.Enabled = True
```

```
End Sub
```

cmdNext_Click() 事件代码如下:

```
Private Sub cmdNext_Click() '“下一条”按钮,移到下一条
```

```
    If Data1.RecordSet.EOF = True Then
```

```
        cmdNext.Enabled = False
```

```
        Data1.RecordSet.MoveLast
```

```
    Else
```

```
        Data1.RecordSet.MoveNext
```

```
    End If
```

```
    cmdPrevious.Enabled = True
```

```
End Sub
```

cmdLast_Click() 事件代码如下:

```
Private Sub cmdLast_Click() '“末一条”按钮,移到最后一条
```

```
    Data1.RecordSet.MoveLast
```

```
    cmdNext.Enabled = False
```

```
    cmdPrevious.Enabled = True
```

```
End Sub
```

4) 保存窗体, 运行程序, 结果如图 10-9 所示。

10.4 数据访问对象及应用

利用 Data 控件和 ADO 控件可以对数据库进行操作, 但 Data 控件和 ADO 控件只给出有限的访问现存数据库的功能。VB 中还提供了通过 DAO 和 ADO 数据访问对象来建立、连接和处理数据库的方法。利用 DAO 和 ADO 对象可以更灵活地完成数据库的各种操作。

10.4.1 DAO 数据访问对象

DAO 是 Data Access Objects 数据访问对象的缩写, 是 Microsoft Jet 数据库引擎的面向对象的接口。它不是可视化的对象, 使用它全部都要靠编码来完成, DAO 是设计关系型数据库系统结构的对象类的集合。它提供了完成管理一个系统所需的全部操作的属性和方法, 包括创建数据库, 定义表、字段和索引, 建立表间的关系, 定位和查询数据库等工具。

1. DAO 对象的结构

DAO 数据访问对象是分层来组织的, 其结构如图 10-11 所示。

顶层为 DBEngine 数据库引擎, 在数据库引擎下是 Workspace 对象集合, Workspace 对象

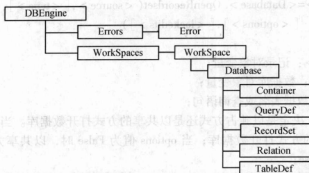


图 10-11 DAO 数据访问对象的结构

包括多个子集。DAO 包括的数据访问对象见表 10-3。

表 10-3 DAO 包括的数据访问对象

对 象	含 义	对 象	含 义
DBEngine	数据库引擎	QueryDef	代表数据库的查询定义
WorkSpace	工作区，包含打开的数据库并提供处理方法	RecordSet	代表数据表或查询中的记录集
Database	代表打开的数据库	Field	代表数据表中的字段
TableDef	代表数据表	Error	数据库访问的出错信息

2. DAO 对象的常用方法

VB 中提供了一些初始化 Database 和 Recordset 对象的方法。通过这些方法就可以操作数据库了。

(1) Set Database 方法

Set Database 方法常用来初始化 Database 对象，格式如下：

```
Set <Database> = <WorkSpace>. OpenDatabase( <dbname> , [ <options> ] ,  
[ <readonly> ] , [ <connect> ] )
```

说明：

- <Database>：Database 对象变量；
- <WorkSpace>：WorkSpace 对象变量；
- <dbname>：数据库文件名；
- <options>：决定是以独占方式还是以共享的方式打开数据库。当 options 值为 True 时，以独占的方式打开数据库；当 options 值为 False 时，以共享方式打开数据库，默认值为 False；
- <readonly>：决定是以只读方式还是以读/写的方式打开数据库。当 readonly 值为 True 时，以只读的方式打开数据库；当 readonly 值为 False 时，以读/写方式打开数据库，默认值为 False；
- <connect>：用来指出数据库类型及打开数据库的口令等，默认值为 Jet 数据库。

(2) Set RecordSet 方法

Set RecordSet 方法常用来初始化 RecordSet 对象，格式如下：



```
Set <RecordSet >= <Database >. OpenRecordset( <source >,[ <type > ],  
[ <options > ],[ <lockedits > ])
```

说明:

- <RecordSet>: 记录对象变量;
- <Database>: 数据库对象变量;
- <source>: 数据表名或查询语句;
- <options>: 决定是以独占方式还是以共享的方式打开数据库。当 options 值为 True 时, 以独占的方式打开数据库; 当 options 值为 False 时, 以共享方式打开数据库, 默认值为 False;
- <type>: 创建记录集的类型; dbOpenTable 表型记录集; dbOpenDynaset 动态型记录集; dbOpenSnapshot 快照型记录集;
- <lockedits>: 指定数据表中的记录不能修改。

(3) RecordSet 对象的方法

参看数据控件的 RecordSet 方法。

3. DAO 对象的引用

要在应用程序中使用 DAO 对象, 必须要添加对 DAO 对象的引用, 方法如下:

1) 在 VB 开发环境中, 依次选择“工程”→“引用”菜单, 打开添加引用窗口, 如图 10-12 所示。

2) 在添加引用窗口中选择 Microsoft DAO 3.51 Object Library, 再单击“确定”按钮, 完成添加 DAO 数据访问对象库的操作。

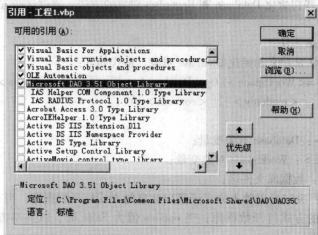


图 10-12 添加 DAO 对象库

10.4.2 ADO 数据访问对象

ADO 数据对象具有灵活而又有效的访问数据库的方式, ADO 已经成为比较常用的访问数据库的方法和手段。

1. ADO 对象的结构

ADO 是 ActiveX Data Objects 的缩写,是微软软件体系中处理关系数据库和非关系数据库的常用技术。ADO 的对象模型如图 10-13 所示。

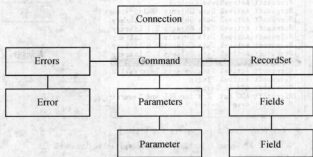


图 10-13 ADO 的对象模型

ADO 中包括的主要数据访问对象如表 10-4 所示。

表 10-4 ADO 包括的主要数据访问对象

对 象	含 义	对 象	含 义
Connection	连接数据库对象	RecordSet	代表数据表或查询中的记录集
Command	操作数据库的指令,可以接收 SQL 指令、表及存储过程	Field	代表数据表中的字段
		Error	数据库访问的出错信息

2. ADO 对象的常用方法

ADO 对象也包括访问数据库的一些方法,具体的方法可参考数据控件和 DAO 对象中方法。

3. ADO 对象的引用

要在应用程序中使用 ADO 对象,必须要添加对 ADO 对象的引用,方法如下:

1) 在 VB 开发环境中,依次选择“工程”→“引用”菜单,打开添加引用窗口,如图 10-14 所示。

2) 在添加引用窗口中选择 Microsoft ActiveX Data Objects 2.8 Library,再单击“确定”按钮,完成添加 ADO 数据访问对象库的操作。

10.4.3 数据访问对象的应用

利用数据访问对象 DAO 和 ADO 可以方便、灵活地对数据库进行各种操作。下面是两个应用的实例。

【例 10-2】 利用 DAO 对象实现例 10-1 中“学生信息”表的操作,包括添加、删除、修改及查询功能。程序运行结果如图 10-15 所示。

操作步骤:

1) 创建窗体及控件,并参照图 10-15 设置控件的相关属性。这里的所有的文本框控件定义为一个控件数组名为 Text1,下标顺序分别为 0~6。

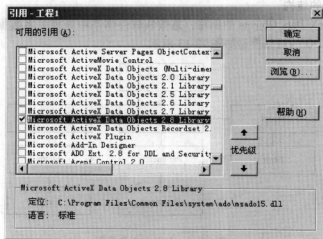


图 10-14 添加 ADO 对象库

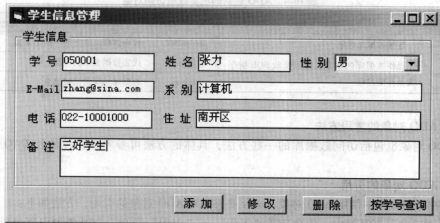


图 10-15 利用 DAO 实现的学生信息管理程序运行结果

2) 添加 DAO 引用 Microsoft DAO 3.51 Object Library。

3) 打开“代码设计”窗口，输入程序代码。

Form_Load() 事件代码如下：

```
Private Sub Form_Load()  
    Combo1.AddItem "男"  
    Combo1.AddItem "女"  
End Sub
```

通用过程 setValue() 的代码：

```
Sub setValue(rs As RecordSet)  
    On Error Resume Next  
    If rs.RecordCount > 0 Then
```

' 根据记录集 rs 设置界面中文本框的值

```

For i = 0 To 1
    Text1(i).Text = rs.Fields(i)
Next i
Combo1.Text = rs.Fields(2)
For i = 2 To 6
    Text1(i).Text = rs.Fields(i + 1)
Next i
End If
End Sub

cmdAdd_Click() 事件代码如下:

Private Sub cmdAdd_Click() '“添加”按钮,添加一条记录
    On Error GoTo errHandler
    Dim db As Database
    Dim rs As RecordSet
    Dim i As Integer
    Dim strtemp As String

    Set db = OpenDatabase(App.Path & "\DBStudent.mdb")
    If Text1(0).Text <> "" Then
        strtemp = "select * from student where 学号=" & Text1(0).Text & ""
        Set rs = db.OpenRecordset(strtemp)
        If rs.RecordCount >= 1 Then
            MsgBox "此学号已经存在!", vbOKOnly, "添加记录"
            Exit Sub
        Else
            Set rs = db.OpenRecordset("select * from student")
            rs.AddNew
            For i = 0 To 1
                rs.Fields(i) = Text1(i).Text
            Next
            rs.Fields(2) = Combo1.Text
            For i = 3 To 7
                rs.Fields(i) = Text1(i - 1).Text
            Next
            rs.Update
            rs.Close
            db.Close
            Set rs = Nothing
            Set db = Nothing
            MsgBox "添加成功!", vbOKOnly, "添加记录"
            Exit Sub
        End If
    End If
End Sub

```

errHandler;

MsgBox "添加记录时出错!", vbCritical, "添加记录"

End Sub

cmdEdit_Click() 事件代码如下:

Private Sub cmdEdit_Click()

' "修改"按钮,修改当前记录

Dim db As Database

Dim rs As RecordSet

Dim i As Integer

Dim strtemp As String

If Text1(0).Text <> "" Then

strtemp = "select * from student where 学号 = " & Text1(0).Text & ""
If MsgBox("是否真的修改?", vbQuestion + vbYesNo, "提示") = vbYes Then

Set db = OpenDatabase(App.Path & "\DBStudent.mdb")

Set rs = db.OpenRecordSet(strtemp)

If rs.RecordCount = 1 Then

rs.Edit

For i = 0 To 1

rs.Fields(i) = Text1(i).Text

Next

rs.Fields(2) = Combo1.Text

For i = 3 To 7

rs.Fields(i) = Text1(i - 1).Text

Next

rs.Update

rs.Close

db.Close

Set rs = Nothing

Set db = Nothing

MsgBox "修改成功!", vbOKOnly, "修改记录"

Exit Sub

Else

MsgBox "不存在", vbCritical, "提示"

Exit Sub

End If

End If

End If

End Sub

cmdDelete_Click() 事件代码如下:

Private Sub cmdDelete_Click()

' "删除"按钮,删除当前记录

Dim db As Database

Dim rs As RecordSet

Dim i As Integer

```

Dim strtemp As String
If Text1(0).Text <> "" Then
    strtemp = "select * from student where 学号='" & Text1(0).Text & "'"
    If MsgBox("是否真的删除?", vbQuestion + vbYesNo, "提示") = vbYes Then
        Set db = OpenDatabase(App.Path & "\DBStudent.mdb")
        Set rs = db.OpenRecordSet(strtemp)
        rs.Delete
        rs.Close
        db.Close
        Set rs = Nothing
        Set db = Nothing
        MsgBox "删除成功!", vbOKOnly, "删除记录"
    End If
End If

End Sub

cmdQuery_Click() 事件代码如下:
Private Sub cmdQuery_Click() ' "查询"按钮,按学号进行查询
    Dim db As Database
    Dim rs As RecordSet
    Dim i As Integer

    strtemp = "select * from student where 学号='" & Text1(0).Text & "'"
    Set db = OpenDatabase(App.Path & "\DBStudent.mdb")
    Set rs = db.OpenRecordSet(strtemp)
    If rs.RecordCount > 0 Then
        Call SetValue(rs)
    End If
    rs.Close
    db.Close
    Set rs = Nothing
    Set db = Nothing
End Sub

```

4) 保存窗体, 运行程序, 结果如图 10-15 所示。

【例 10-3】 利用 ADO 对象实现对 SQL Server2000 中数据库: DBStudent 中的表, 即学生信息表进行操作, 包括添加、删除、修改及查询功能。程序运行结果见例 10-2 中的图 10-15 所示。已知, SQL Server2000 数据库服务器名为 DBTest; 用户名为 sa; 密码为 1234。

操作步骤:

1) 在 SQL Server2000 中建立数据库: DBStudent, 并建立表 10-5 所示的学生信息表。



表 10-5 学生信息表

字段名	字段类型	字段大小	是否为空	字段名	字段类型	字段大小	是否为空
学号	varchar	6	否(主键)	系别	varchar	20	是
姓名	varchar	10	否	住址	varchar	50	是
性别	varchar	2	否	电话	varchar	12	是
E_Mail	varchar	20	是	备注	varchar	100	是

2) 创建窗体及控件,并参照图 10-15 设置控件的相关属性。这里的所有的文本框控件定义为一个控件数组名为 Text1,下标顺序分别为 0~6。

3) 添加 ADO 引用 Microsoft ActiveX Data Objects 2.8 Library。

4) 打开“代码设计”窗口,输入程序代码。

建立窗体级变量 conn,并建立 OpenConnection 过程,用于打开数据库连接,代码如下:

```

Dim conn As New ADODB.Connection '声明 Connection 对象 conn
Sub OpenConnection() '用于打开数据库连接,即初始化对象 conn
    Set conn = New ADODB.Connection
    With conn
        .ConnectionString = "Provider = SQLOLEDB;User ID = sa;Password = 1234;
        Initial Catalog = DBStudent;Data Source = 127.0.0.1"
        .CursorLocation = adUseClient
        .Open
    End With
End Sub

```

通用过程 setValue() 的代码如下:

```

Sub setValue(rs As RecordSet) '根据记录集 rs 设置界面中文本框的值
    On Error Resume Next
    If rs.RecordCount > 0 Then
        For i = 0 To 1
            Text1(i).Text = rs.Fields(i)
        Next i
        Combo1.Text = rs.Fields(2)
        For i = 2 To 6
            Text1(i).Text = rs.Fields(i + 1)
        Next i
    End If
End Sub

```

Form_Load() 事件代码如下:

```

Private Sub Form_Load()
    Combo1.AddItem "男"
    Combo1.AddItem "女"
End Sub

```

cmdAdd_Click() 事件代码如下:

```

Private Sub cmdAdd_Click() '“添加”按钮,添加一条记录
    On Error GoTo errHandler
    Dim strSql As String
    Dim rs As New ADODB.RecordSet
    Dim strtemp As String

    If Text1(0).Text <> "" Then
        strtemp = "select * from 学生信息 where 学号='" & Text1(0).Text & "'"
        Call OpenConnection
        Set rs = conn.Execute(strtemp)
        If rs.RecordCount >= 1 Then
            MsgBox "此学号已经存在!", vbOKOnly, "添加记录"
            Exit Sub
        Else
            strSql = "insert into 学生信息(学号,姓名,性别,E_Mail,系别,电话,住址,备注) values('"

            strSql = strSql & Text1(0).Text & "','"
            strSql = strSql & Text1(1).Text & "','"
            strSql = strSql & Comb1.Text & "','"
            strSql = strSql & Text1(2).Text & "','"
            strSql = strSql & Text1(3).Text & "','"
            strSql = strSql & Text1(4).Text & "','"
            strSql = strSql & Text1(5).Text & "','"
            strSql = strSql & Text1(6).Text & "')"
            conn.Execute(strSql)
            MsgBox "添加成功!", vbOKOnly, "添加记录"
            Exit Sub
        End If
        conn.Close
        Set conn = Nothing
    End If
errHandler:
    MsgBox "添加记录时出错!", vbCritical, "添加记录"
End Sub

cmdEdit_Click() 事件代码如下:
Private Sub cmdEdit_Click() '“修改”按钮,修改当前记录
    Dim rs As RecordSet
    Dim i As Integer
    Dim strSql As String

    If Text1(0).Text <> "" Then
        If MsgBox("是否真的修改?", vbQuestion + vbYesNo, "提示") = vbYes Then
            strSql = "update 学生信息 set "

```



```

strSql = strSql & "姓名='" & Text1(1).Text & "',"
strSql = strSql & "性别='" & Comb1.Text & "',"
strSql = strSql & "E_Mail='" & Text1(2).Text & "',"
strSql = strSql & "系别='" & Text1(3).Text & "',"
strSql = strSql & "电话='" & Text1(4).Text & "',"
strSql = strSql & "住址='" & Text1(5).Text & "',"
strSql = strSql & "备注='" & Text1(6).Text & "'"
strSql = strSql & " where 学号='" & Text1(0).Text & "'"
Call OpenConnection
conn.Execute(strSql)
conn.Close
Set conn = Nothing
MsgBox "修改成功!", vbOKOnly, "修改记录"
End If

```

```
End If
```

```
End Sub
```

cmdDelete_Click() 事件代码如下:

```

Private Sub cmdDelete_Click() '“删除”按钮,删除当前记录
Dim strtemp As String
If Text1(0).Text <> "" Then
strtemp = "delete from 学生信息 where 学号='" & Text1(0).Text & "'"
If MsgBox("是否真的删除?", vbQuestion + vbYesNo, "提示") = vbYes Then
Call OpenConnection
conn.Execute(strtemp)
For i = 0 To 6
Text1(i).Text = ""
Next i
Comb1.Text = ""
conn.Close
Set conn = Nothing
MsgBox "删除成功!", vbOKOnly, "删除记录"
End If
End If
End Sub

```

cmdQuery_Click() 事件代码如下:

```

Private Sub cmdQuery_Click() '“查询”按钮,按学号进行查询
Dim rs As New ADODB.RecordSet
Dim i As Integer
strtemp = "select * from 学生信息 where 学号='" & Text1(0).Text & "'"
OpenConnection
Set rs = conn.Execute(strtemp)
If rs.RecordCount > 0 Then

```

```

Call setValue(rs)
Else
    MsgBox "此学号不存在!", vbCritical, "提示"
End If
rs.Close
Set rs = Nothing
conn.Close
Set conn = Nothing
End Sub

```

5) 保存窗体，运行程序，结果如图 10-15 所示。

习 题

- 10-1 在 VB 中访问数据库的控件和对象有哪些？
 10-2 关系数据库的主要功能有哪些？
 10-3 简述基本的 SQL 语句有哪些？主要功能是什么？
 10-4 数据绑定控件有哪些？
 10-5 ADO 中包括哪些对象？并分别描述它们的功能。
 10-6 在 Access 中建立一个数据库：DBStudent.mdb；其中包含表：Student。其结构见表 10-6。

表 10-6 习题 10-6 的表

字 段	类 型	大 小	是否为空	字 段	类 型	大 小	是否为空
学号	Text	6	否(主键)	数学	Single	8	否
姓名	Text	10	否	语文	Single	8	否
年龄	Integer	4	是	英语	Single	8	否
性别	Text	2	是				

程序要求：

- 1) 设计一个窗体，编写程序能够对 DBStudent.mdb 数据库中 Student 表进行编辑、添加、删除等操作。
- 2) 设计一个窗体，编写程序浏览学生基本信息，查询某指定学生考试成绩。
- 3) 建议使用如下几种方式分别来实现：
 - Data 控件和数据绑定控件实现；
 - 通过使用 DAO 对象及控件实现；
 - 使用 ADO Data 控件和数据绑定控件实现；
 - 使用 ADO 对象及控件实现。

10-7 在 SQL Server2000 中建立习题 10-6 中的数据库：DBStudent；表：Student。利用 ADO 对象实现习题 10-6 的功能。

第 11 章 菜单、工具栏与状态栏

用户界面是一个应用程序的重要组成部分。本章将介绍在 VB 中用户界面设计的工具和方法,包括菜单、工具栏、状态栏等知识。

菜单、工具栏及状态栏在应用程序的设计中有着重要的作用,一个应用程序,若能设计出可供有效使用和简捷操作的菜单、工具栏,可使其程序的质量得到很大的提升。

11.1 菜单

菜单是 Windows 应用程序中十分关键的要素之一。它可以将命令以分组的形式显示,为用户灵活操作应用程序提供了便捷的手段。

在实际的应用中,应用程序的菜单可以分为两种基本类型:下拉式菜单和上下文菜单。

11.1.1 菜单的构成

菜单通常由一些相关的命令项组成。一个完整的菜单包括多个菜单项,每一菜单项是一个语句,用来实现某一具体的操作任务。

下拉菜单是由菜单栏、菜单标题项、菜单项和子菜单项组成的,而上下文菜单是由菜单项和子菜单组成的。表 11-1 列出了菜单中常见的构成项及含义。

表 11-1 菜单中常见的构成项及含义

菜单的构成项	说 明
菜单栏	用于放置多个菜单的标题
菜单标题项	是每个菜单的名称,是一个菜单列表项的首菜单项,如文件等
菜单项	也叫命令项,在主菜单列出的选项都叫菜单项
子菜单	属于某个菜单项的下一级菜单

图 11-1 和图 11-2 分别列出了 VB 应用程序的下拉菜单和上下文菜单。

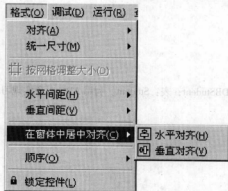


图 11-1 VB 的下拉菜单

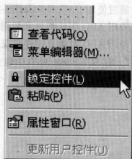


图 11-2 VB 的上下文菜单

11.1.2 下拉菜单

下拉菜单是 Windows 应用程序最常用的一种菜单形式。菜单通过菜单编辑器创建。

1. 菜单编辑器

在 VB 系统环境下,通过依次选择“工具”→“菜单编辑器”菜单选项,打开“菜单编辑器”窗口,如图 11-3 所示。

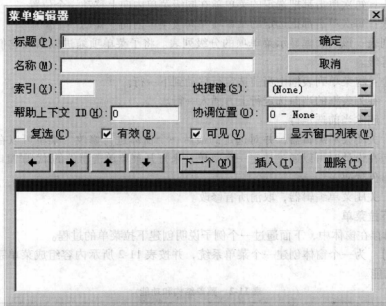


图 11-3 “菜单编辑器”窗口

菜单编辑器的各个选项包括:

- 标题: 用该选项可以输入菜单名或命令名, 这些名字出现在菜单条或菜单之中。
- 名称: 允许为菜单输入的名称, 是菜单的标识, 只用于访问代码中的菜单项, 不会出现在菜单中。
- 索引: 为一整型值, 在建立菜单控件数组时, 指定该项菜单控件数组的下标。在菜单控件数组中, 其索引值可以不连续。
- 快捷键: 允许为每个命令选定快捷键。
- 帮助上下文 ID: 允许为 Context ID 指定唯一数值。在 HelpFile 属性指定的帮助文件中用该数值查找适当的帮助主题。
- 协调位置: 允许选择菜单的 NegotiatePosition 属性。该属性决定是否及如何在容器窗体中显示菜单。
- 复选: 允许在菜单项的左边设置复选标记。通常用它来指出切换选项的开关状态。
- 有效: 由此选项可决定是否让菜单项对事件作出响应, 而如果希望该项失效并模糊显示出来, 则也可清除事件。
- 可见: 设置菜单项是否显示在菜单上。



- 显示窗口列表：在 MDI 应用程序中，确定菜单控件是否包含一个打开的 MDI 子窗体列表。
- 右箭头：每次单击都把选定的菜单向右移一个等级。一共可以创建 4 个子菜单等级。
- 左箭头：每次单击都把选定的菜单向左移一个等级。一共可以创建 4 个子菜单等级。
- 上箭头：每次单击都把选定的菜单项在同级菜单内向上移动一个位置。
- 下箭头：每次单击都把选定的菜单项在同级菜单内向下移动一个位置。
- 菜单列表：该列表框显示菜单项的分级列表。将子菜单项缩进以指出它们的分级位置或等级。
- 下一个：将菜单列表框中当前选定行移动到下一行。
- 插入：在列表框的当前选定行上方插入一行。
- 删除：删除当前选定行。
- 确定：关闭菜单编辑器，并对选定的最后一个窗体进行修改。菜单可以在设计时使用，但在设计时可以通过选定一个菜单，来打开菜单单击事件的“代码”窗口，而不是执行事件代码。
- 取消：关闭菜单编辑器，取消所有修改。

2. 创建下拉菜单

下拉菜单存在窗体中，下面通过一个例子说明创建下拉菜单的过程。

【例 11-1】 为一个窗体创建一个菜单系统，并按表 11-2 所示内容组成菜单系统的各级菜单选项及功能。

表 11-2 菜单结构和功能

菜单名	子菜单	菜单功能描述
文件	新建	新建文本文件
	打开	打开文本文件
	保存	保存文本文件
	退出	关闭程序
编辑	复制	复制选中的文本
	剪切	剪切选中的文本
	粘贴	粘贴内容到当前位置
	删除	删除选中的文本
格式	字体	打开字体设置对话框
	背景色	打开颜色设置对话框

操作步骤：

- 1) 创建一个窗体，打开菜单编辑器。
- 2) 设计主菜单，在菜单编辑器中，设计菜单标题、名称及热键。如图 11-4 所示。
- 3) 设计子菜单：在菜单编辑器中，按表 11-2 设计子菜单标题、名称、热键及快捷键。如图 11-5 所示。

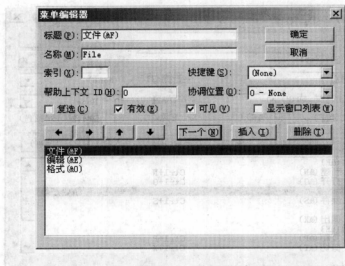


图 11-4 设计主菜单

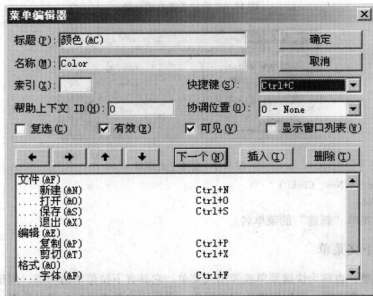


图 11-5 设计子菜单

4) 为菜单项分组：本例的菜单中包括多个子菜单，将菜单项按功能分组可加快用户查找相应菜单的速度，常用的菜单分组方式是在菜单项之间加上分割条“-”。如图 11-6 所示。

5) 运行程序，窗体上的菜单如图 11-7 所示。

3. 菜单的使用

菜单的使用主要通过生成菜单的单击事件即 Click 事件来实现的。例如，例 11-1 中为菜单“新建”创建单击事件，格式如下：



图 11-6 设计子菜单项分组

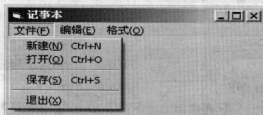


图 11-7 窗体上的菜单

```
Private Sub New_Click()  
End Sub
```

其中, New 为菜单“新建”的菜单名。

11.1.3 上下文菜单

上下文菜单,也称为快捷菜单或弹出式菜单。它具有下拉菜单的功能,但比下拉菜单使用起来更灵活、更方便,可以跟踪用户的操作。根据用户单击鼠标右键的位置,动态地调整菜单项的显示位置。

1. 上下文菜单创建

创建上下文菜单的方法与下拉菜单的创建方式基本相同,它不需要顶部下拉,因此在设计菜单时,菜单标题要设置成不可见的。如图 11-8 所示。

2. 上下文菜单的显示

上下文菜单通过调用 PopupMenu 方法显示,PopupMenu 方法的格式如下:

```
[ <对象> . ] PopupMenu <上下文菜单名> [ , <标志> ] [ , x ] [ , y ]
```

功能: 在对象的指定位置上显示以 <上下文菜单名> 为名的菜单。

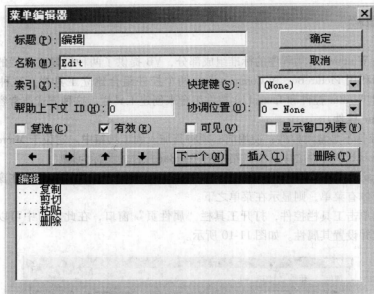


图 11-8 创建上下文菜单

3. 应用

上下文菜单的显示一般通过单击鼠标右键弹出的。具体实现请看下面的例子。

【例 11-2】 在一个窗体上单击鼠标右键显示如图 11-8 所示的上下文菜单。

操作步骤如下：

- 1) 按图 11-8 所示创建上下文菜单 Edit。
- 2) 生成窗体的 `Form_MouseDown()` 事件，并调用 `PopupMenu` 方法。代码如下：

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 2 Then '2 为鼠标右键
        PopupMenu Edit
    End If
End Sub
```

- 3) 运行程序，在窗体上单击鼠标右键，弹出上下文菜单 Edit。如图 11-9 所示。

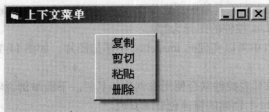



图 11-9 窗体上的上下文菜单

11.2 工具栏

工具栏是 Windows 应用程序的标准组成部分, VB 提供了两种创建工具栏的方法: 一是手工制作, 即利用 PictureBox 和 CommandBox 两个控件组合建立工具栏, 这种方法比较繁琐; 二是利用 ToolBar 控件和 ImageList 控件组合来创建工具栏, 这种方式非常容易和方便。

1. ToolBar 控件

ToolBar 控件不是常用控件, 使用前必须将其加入到工具箱中。它位于 Microsoft Windows Common Controls 6.0 部件之中, 工具箱中的 ToolBar 控件的图标为 。

双击工具箱上 ToolBar 控件为窗体添加一个新的工具栏, 工具栏会显示在窗体上的标题栏之下。若窗体有菜单, 则显示在菜单之下。

鼠标右键单击工具栏控件, 打开工具栏“属性页”窗口, 在此窗口中可以设置工具栏中包含的按钮并设置其属性。如图 11-10 所示。

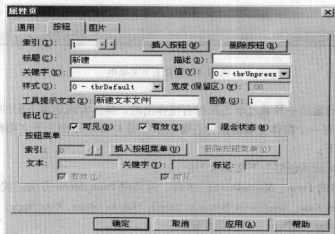



图 11-10 ToolBar 控件的属性页

2. ImageList 控件

ImageList 控件用来保存显示在 ToolBar 控件中按钮的图片, ImageList 控件不是常用控件, 使用前必须将其加入到工具箱中。它位于 Microsoft Windows Common Controls 6.0 部件之中, 工具箱中的 ImageList 控件的图标为 。

在“属性页”窗口中可以实现向 ImageList 中添加图片。如图 11-11 所示。

3. 工具栏应用

工具栏通常和图像列表控件联合使用来创建工具栏。下面举例说明创建工具栏的过程。

【例 11-3】 为例 11-1 创建的窗体创建工具栏。

操作步骤如下:

1) 向窗体上添加 ImageList 控件, 打开其“属性页”窗口, 为 ImageList 控件添加所需的图像。如图 11-11 所示。

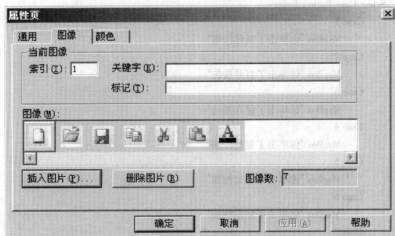


图 11-11 ImageList 控件的属性页

2) 向窗体上添加 ToolBar 控件, 打开其“属性页”窗口, 在“通用”选项卡中指定 ToolBar 所关联的 ImageList 控件。如图 11-12 所示。

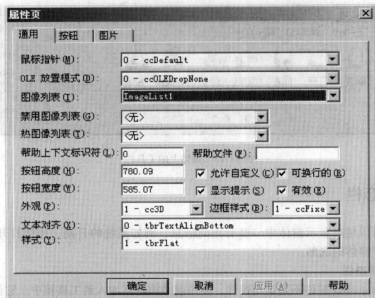


图 11-12 设置 ToolBar 的图像列表控件

向 ToolBar 控件添加多个 Button 对象, 并定义它们的标题和图像。如图 11-10 所示。

(3) 为工具栏中的按钮添加响应代码, 可以编写 ToolBar 控件的 ButtonClick() 事件程序。并通过 Button.Index 属性值来判断单击了哪一个按钮, 一般通过 Select Case 结构进行相应的处理。具体的程序如下:

```
Private Sub ToolBar1_ButtonClick(ByVal Button As MSCOMctlLib.Button)
```



```
Select Case Button.Index
```

```
Case 1
```

```
MsgBox "你单击了新建按钮"
```

```
Case 2
```

```
MsgBox "你单击了打开按钮"
```

```
Case 3
```

```
MsgBox "你单击了保存按钮"
```

```
Case 4
```

```
MsgBox "你单击了复制按钮"
```

```
Case 5
```

```
MsgBox "你单击了剪切按钮"
```

```
Case 6
```

```
MsgBox "你单击了粘贴按钮"
```

```
Case 7
```

```
MsgBox "你单击了字体按钮"
```

```
End Select
```

```
End Sub
```

4) 运行程序，窗体上的工具栏如图 11-13 所示。

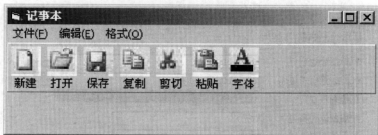


图 11-13 窗体上的工具栏

11.3 状态栏

状态栏可以用来显示窗体中一些有用的信息，例如系统的日期、当前打开文件的名称等，一般位于窗体的底部。

1. 创建状态栏

状态栏 StatusBar 控件不是常用控件，使用前必须将其加入到工具箱中。它位于 Microsoft Windows Common Controls 6.0 部件之中，工具箱中的 StatusBar 控件的图标为

双击工具箱中 StatusBar 控件，就可以将状态栏加入到窗体上了。

2. 设计状态栏

设计状态栏是通过状态栏的属性进行设置的。打开状态栏的“属性页”，如图 11-14 所示。选择“窗格”选项卡，就可以进行状态栏的设计了。

3. 应用

由于状态栏经常要显示应用程序的某些当前的状态，因此有些窗格的内容需要在运行时



图 11-14 StatusBar 控件属性页

进行设置。

下面举例说明 StatusBar 控件的应用。

【例 11-4】 为例 11-1 创建的窗体创建状态栏，并在状态栏上显示系统的日期和时间。
操作步骤如下：

- 1) 在例 11-1 所示的窗体上添加 StatusBar 控件，并在其上设置两个窗格。
- 2) 添加 Timer 控件，并生成 Timer 控件的 Timer() 事件。代码如下：

```
Private Sub Timer1_Timer()  
    StatusBar1.Panels(1).Text = "当前系统日期及时间:"  
    StatusBar1.Panels(2).Text = Now  
End Sub
```

- 3) 运行程序，窗体上的状态栏及其内容如图 11-15 所示。

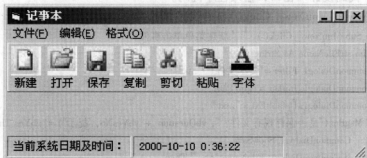


图 11-15 窗体上的状态栏

11.4 综合应用实例

【例 11-5】 利用 TextBox 控件及菜单、工具栏、状态栏等创建一个简易的文本编辑器。如图 11-16 所示。

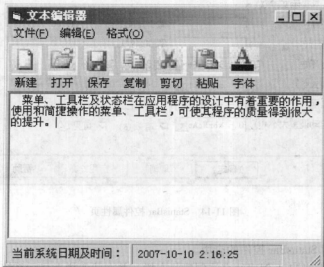



图 11-16 创建文本编辑器

操作步骤如下：

- 1) 按例 11-1 和例 11-3 创建菜单和工具栏。
- 2) 按例 11-4 创建状态栏。
- 3) 添加控件 CommonDialog，其在 Microsoft Common Dialog Controls 6.0 中。加入后在工具栏中的图标为 .

4) 文本编辑器窗体及控件如图 11-17 所示。

5) 打开“代码设计”窗口，输入程序代码。

菜单 MnuNew 的 MnuNew_Click() 事件代码如下：

```
Private Sub MnuNew_Click() ' 新建菜单单击事件
    Dim strFileName As String
    CommonDialog1.Filter = "文本文件|*.txt"
    CommonDialog1.InitDir = App.Path
    CommonDialog1.DefaultExt = ".txt"
    If MsgBox("是否保存现有文件?", vbQuestion + vbYesNo, "提示") = vbYes Then
        CommonDialog1.ShowSave
        If CommonDialog1.FileName <> "" Then
            strFileName = CommonDialog1.FileName
            Open strFileName For Output As #1
            Print #1, txtArea.Text
```



图 11-17 文本编辑器窗体及控件

```

Close #1
End If
End If
txtArea.Text = ""
End Sub

```

菜单 MnuOpen 的 MnuOpen_Click() 事件代码如下:

```

Private Sub MnuOpen_Click() ' 打开菜单单击事件
    Dim strFileName As String
    Dim strTemp As String
    CommonDialog1.Filter = "文本文件|*.txt"
    CommonDialog1.InitDir = App.Path
    CommonDialog1.DefaultExt = ".txt"
    CommonDialog1.ShowOpen
    If CommonDialog1.FileName <> "" Then
        txtArea.Text = ""
        strFileName = CommonDialog1.FileName
        Open strFileName For Input As #1
        Do While Not EOF(1)
            Line Input #1, strTemp
            txtArea.Text = txtArea.Text & strTemp & vbCrLf
        Loop
    End If
    Close #1
End Sub

```

```
End If
```

```
End Sub
```

菜单 MnuSave 的 MnuSave_Click() 事件代码如下:

```
Private Sub MnuSave_Click() '保存菜单单击事件
```

```
Dim strFileName As String
```

```
CommonDialog1.Filter = "文本文件|*.txt"
```

```
CommonDialog1.InitDir = App.Path
```

```
CommonDialog1.DefaultExt = ".txt"
```

```
CommonDialog1.ShowSave
```

```
If CommonDialog1.FileName <> "" Then
```

```
strFileName = CommonDialog1.FileName
```

```
Open strFileName For Output As #1
```

```
Print #1, txtArea.Text
```

```
Close #1
```

```
End If
```

```
End Sub
```

菜单 MnuCopy 的 MnuCopy_Click() 事件代码如下:

```
Private Sub MnuCopy_Click() '复制菜单单击事件
```

```
Dim strTemp As String
```

```
strTemp = txtArea.Text
```

```
If txtArea.Text <> "" Then
```

```
Clipboard.Clear
```

```
Clipboard.SetText strTemp, vbCFRTF
```

```
MnuPaste.Enabled = True
```

```
ToolBar1.Buttons(5).Enabled = True
```

```
End If
```

```
End Sub
```

菜单 MnuCut 的 MnuCut_Click() 事件代码如下:

```
Private Sub MnuCut_Click() '剪切菜单单击事件
```

```
If txtArea.Text <> "" Then
```

```
Call MnuCopy_Click
```

```
End If
```

```
txtArea.Text = ""
```

```
End Sub
```

菜单 MnuPaste 的 MnuPaste_Click() 事件代码如下:

```
Private Sub MnuPaste_Click() '粘贴菜单单击事件
```

```
txtArea.Text = Clipboard.GetText
```

```
End Sub
```

菜单 MnuFont 的 MnuFont_Click() 事件代码如下:

```
Private Sub MnuFont_Click() '字体菜单单击事件
```

```
CommonDialog1.ShowFont
```

```
txtArea.FontBold = CommonDialog1.FontBold
```

```

txtArea.FontItalic = CommonDialog1.FontItalic
txtArea.FontSize = CommonDialog1.FontSize
txtArea.FontStrikethru = CommonDialog1.FontStrikethru
txtArea.FontUnderline = CommonDialog1.FontUnderline
txtArea.ForeColor = CommonDialog1.Color

```

End Sub

菜单 MnuColor 的 MnuColor_Click() 事件代码如下:

```

Private Sub MnuColor_Click() '颜色菜单单击事件
    CommonDialog1.ShowColor
    txtArea.BackColor = CommonDialog1.Color

```

End Sub

工具栏 ToolBar1 按钮单击事件 ToolBar1_ButtonClick() 代码如下:

```

Private Sub ToolBar1_ButtonClick(ByVal Button As MSCOMctlLib.Button)
    Select Case Button.Index
        Case 1
            Call MnuNew_Click
        Case 2
            Call MnuOpen_Click
        Case 3
            Call MnuSave_Click
        Case 4
            Call MnuCopy_Click
        Case 5
            Call MnuCut_Click
        Case 6
            Call MnuPaste_Click
        Case 7
            Call MnuFont_Click
    End Select

```

End Sub

菜单 MnuExit 的 MnuExit_Click() 事件代码如下:

```

Private Sub MnuExit_Click() '退出菜单单击事件
    Unload Me
End Sub

```

6) 运行程序, 结果如图 11-16 所示。

习 题

- 11-1 菜单、工具栏、状态栏的作用是什么?
- 11-2 菜单分为哪几类? 通常由哪几部分构成?
- 11-3 创建工具栏由哪两个控件组成? 它们的作用是什么?
- 11-4 创建工具栏的步骤是什么?

11-5 如何在状态栏中显示相关信息?

11-6 编写程序:

设计一个窗体,要求带有菜单、工具栏和状态栏,实现在窗体上显示可移动的文本,并可通过菜单或工具栏按钮设置文本的字体信息,如粗体、字号、斜体、颜色等。在状态栏中实时显示当前系统日期及时间。可通过按钮控制文本是否移动。

第 12 章 MDI 窗体

用户界面是一个应用程序的重要组成部分。对用户来讲，界面就是应用程序，感觉不到后台执行的代码。大部分应用程序的主界面都采用 MDI 窗体的形式。

12.1 用户界面概述

用户界面的样式主要有两种：单文档界面(SDI, Single Document Interface)和多文档界面(MDI, Multiple Document Interface)。

单文档界面的一个示例就是 Microsoft Windows 中的 WordPad (写字板) 应用程序。在 WordPad 中，只能打开一个文档，想要打开另一个文档时，必须先关上已打开的文档。如图 12-1 所示。

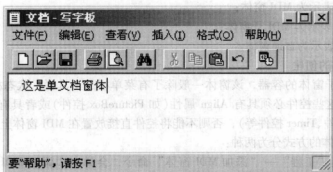


图 12-1 单文档界面应用程序——写字板

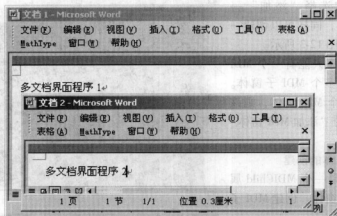


图 12-2 多文档界面应用程序——Word



绝大多数 Windows 应用程序都是多文档界面的形式,如 Microsoft Excel、Microsoft Access 和 Microsoft Word 等都是多文档界面的应用程序。多文档界面的应用程序可以同时打开多个文档。如图 12-2 所示。

12.2 MDI 窗体的构成和创建


1. MDI 窗体的构成


MDI 窗体通常由父窗口和子窗口组成,一个父窗口可包含多个子窗口。子窗口最小化后以图标形式出现在父窗口中,而不会出现在 Windows 的任务栏中;当最小化父窗口时,所有的子窗口也被最小化,但只有父窗口的图标出现在任务栏中。


在 VB 中,父窗口就是 MDI 窗体,子窗口是指 MDIChild 属性为 True 的普通窗口。通过查看窗体的 MDIChild 属性或检查工程资源管理器,可以确定窗体是一个 MDI 窗体还是一个 MDI 子窗体。

1) 若 MDIChild 属性为 True,则它是一个子窗体。

2) 根据图标判断:

■ 图标 : 表示为 MDI 窗体;

■ 图标 : 表示为 MDI 子窗体;

■ 图标 : 表示为普通窗体。

2. MDI 窗体的创建

MDI 窗体是子窗体的容器,该窗体一般除了有菜单栏、工具栏、状态栏外,还可以包括其他控件,但这些控件必须具有 Align 属性(如 PictureBox 控件)或者具有不可见界面(如 CommonDialog 控件、Timer 控件等),否则不能将控件直接放置在 MDI 窗体上。

创建 MDI 窗体的方式分为两种:

1) 依次选择“工程”→“添加 MDI 窗体”命令,会弹出“添加 MDI 窗体”对话框;单击“打开”按钮,即可以创建一个 MDI 窗体。

2) 在工程资源管理器中,单击鼠标右键,依次选择“添加”→“添加 MDI 窗体”,同样可以创建一个 MDI 窗体。如图 12-3 所示。

一个应用程序只能有一个 MDI 窗体,但可以有多子 MDI 子窗体。若工程中已有一个 MDI 窗体,则该“工程”菜单中的“添加 MDI 窗体”子菜单不可用。

3. MDI 子窗体的创建

MDI 子窗体是一个 MDIChild 属性为 True 的普通窗体。创建 MDI 子窗体的步骤如下:

1) 首先创建一个普通窗体。

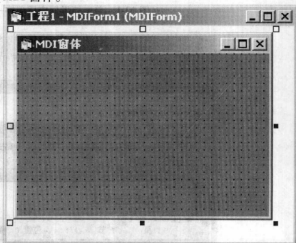


图 12-3 创建 MDI 窗体

2) 修改普通窗体的属性 MDIChild, 将其值设为 True, 即创建了一个 MDI 子窗体。如图 12-4 所示。

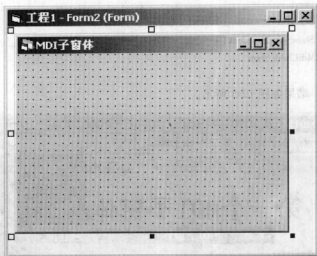


图 12-4 创建 MDI 子窗体

若要创建多个 MDI 子窗体, 重复上述操作即可。

12.3 MDI 窗体的应用

MDI 窗体的使用方法, 一般通过在 MDI 窗体上建立菜单和工具栏, 利用菜单或工具栏对 MDI 子窗体进行操作。

1. MDI 窗体和 MDI 子窗体的显示

显示 MDI 窗体和 MDI 子窗体的方法为 Show, 调用方式如下:

窗体名.Show

一般将此代码放在菜单或工具栏的按钮单击事件中, 从而显示 MDI 窗体或 MDI 子窗体。

当加载 MDI 子窗体时, 其父窗体(MDI 窗体)会自动加载并显示; 而加载 MDI 父窗体时, 其 MDI 子窗体并不会自动加载。

2. 应用举例

【例 12-1】 创建一个 MDI 窗体和两个 MDI 子窗体, 通过单击 MDI 窗体上的菜单来分别显示 MDI 子窗体。

操作步骤如下:

- 1) 创建一个 MDI 窗体和两个 MDI 子窗体。
- 2) 利用“菜单编辑器”在 MDI 窗体上创建菜单。
- 3) 设计两个 MDI 窗体, 在 MDI 子窗体上用 Label 标签控件分别显示不同的文本信息。
- 4) 在“代码设计”窗口, 编写 MDI 窗体菜单事件代码。

菜单 ShowFirst 的 ShowFirst_Click() 事件代码如下:

```
Private Sub ShowFirst_Click() 单击窗体时，显示第一个MDI子窗体
```

```
FirstMdiChild.Show
```

```
End Sub
```

菜单 ShowFirst 的 ShowSecond_Click() 事件代码如下：

```
Private Sub ShowSecond_Click()
```

```
SecondMdiChild.Show
```

```
End Sub
```

5) 运行程序，结果如图 12-5 所示。

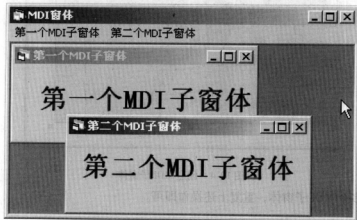


图 12-5 MDI 窗体运行结果

习 题

- 12-1 试述窗体、MDI 窗体、MDI 子窗体之间的区别。
- 12-2 MDI 子窗体能否单独作为启动窗体？当关闭 MDI 子窗体时，其 MDI 窗体是否随之关闭？
- 12-3 关闭 MDI 窗体其 MDI 子窗体是否随之关闭？
- 12-4 MDI 窗体可以放置哪些控件？
- 12-5 如何创建一个 MDI 子窗体？如何显示一个 MDI 子窗体？
- 12-6 编写程序：
 - (1) 设计一个 MDI 窗体；
 - (2) 在此窗体上添加菜单、工具栏及状态栏控件；
 - (3) 实现第 11 章中图 11-16 所示的文本编辑器功能。

第 13 章 程序调试与部署

对于编程人员来说,无论是初学者还是经验丰富的程序员,不可避免地会出现这样或那样的错误。查找和修改错误的过程称为程序的调试。调试是应用程序开发中不可缺少的过程。VB 中提供了一组有效的调试工具,使用这些调试工具,可以快捷地检查程序中错误产生的原因。

当一个程序调试结束,交给用户使用时,一般不会提供源代码。这时,需要生成一个安装文件来部署应用程序,以脱离开发环境运行。

13.1 常见错误

VB 将错误分为编译错误、运行时错误和逻辑错误 3 种。

1. 编译错误

编译错误是指使得 VB 的编译器无法对代码进行编译的错误。如果一个过程中包含了编译错误,VB 将不执行该过程,并且,编程人员也不能向用户提供带有编译错误的运行期版本应用程序。大多数编译错误都是由于违反了 VB 的有关语法而产生的错误,常见的错误包括:

- 1) 标点符号错误,VB 中支持西文标点。
- 2) 关键字拼写错误,如将 Dim 写成 Din。
- 3) 语句行结束符错误。
- 4) 对象未定义或写错,如标签控件的 name 属性为 Label1,使用时写成了 Label2。
- 5) 语句的格式出错,如 If 语句、For 语句等。

对这类错误,VB 可以设置自动语法检测,在代码编写完毕后,错误行会以红色显示,并弹出如图 13-1 所示的出错提示窗口。

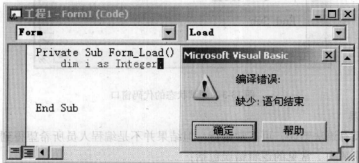


图 13-1 编辑时的出错提示窗口



2. 运行时错误

应用程序在运行期间，当一个语句试图执行一个不能执行的操作时，就会产生运行时错误。常见的错误包括：

- 1) 除法运算时除数为零。
- 2) 连接数据库时，数据库服务器未处于运行状态。
- 3) 被操作的驱动器未准备好或磁盘读/写有错。
- 4) 为变量赋值时，超出了数据的表示范围，例如，Integer 类型表示范围为 -32768 ~ 32767；若为一个 Integer 类型变量赋值 40000，则产生溢出错误。
- 5) 使用数组时下标越界、数据类型不符等。

在运行程序时，VB 会检测到运行时错误，并将弹出一个出错子窗口，提示出错信息。如图 13-2 所示。

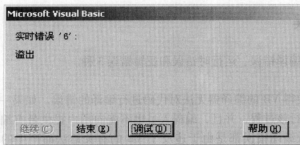


图 13-2 运行时错误提示窗口

在图 13-2 中，单击“帮助”按钮可了解出错信息；单击“调试”按钮可进入中断调试模式，引起错误的语句反色显示。此时可以修改代码，如图 13-3 所示。

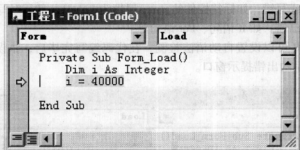


图 13-3 调试状态的代码窗口

3. 逻辑错误

这种错误没有语法错误，可正常运行，但结果并不是编程人员所希望得到的结果。这类错误一般不容易解决。常见的逻辑错误包括：

- 1) 运算符用错。
- 2) If 语句、For 语句、While 语句等的条件出错。
- 3) 一个项目的需求分析错误等。

这类错误需要程序员查找源代码进行改正, 相比较而言, 解决此类问题较难。

13.2 规范化编程

规范化程序代码是一种预防性措施, 采用正确合理的代码书写风格, 能保证代码有较高的可读性和可修改性, 在出现错误时能很快地找到原因。规范化代码时, 最重要的是保持一致性, 也就是说使用同样的要求、布局样式来书写代码。

注释语句虽然本身不运行, 但对提高代码的可读性有重要意义。除了注释以外, 还应该使用统一的命名方式, 即使用统一的命名规则来为每个组件和变量命名。在 VB 中, 用户可以通过设置一些规则来规范化程序代码, 可以使用文本编辑器工具栏对代码执行不同的操作。

文本编辑器工具栏是编辑代码和规范化代码的重要工具。通过使用该工具栏, 可以帮助编程人员快速建立结构化代码。当鼠标指向工具栏中某按钮时, 屏幕上将显示出该工具的提示信息。如图 13-4 所示。



图 13-4 “编辑”工具栏

可以通过选项窗口中的“编辑器格式”选项卡中的选项设置代码的字体、颜色等样式。要打开“编辑器格式”选项卡界面, 依次选择“工具”→“选项”菜单, 再在打开的“选项”对话框中选择“编辑器格式”选项卡, 如图 13-5 所示。

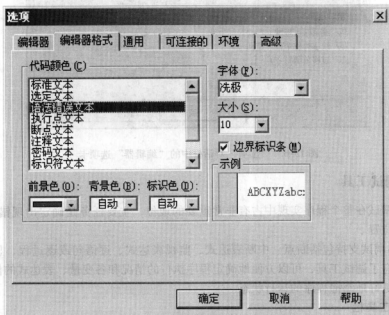


图 13-5 选项窗口中的“编辑器格式”选项卡

在此界面就可以根据自己需要设置相应的选项信息了。

13.3 Visual Basic 中的调试工具

为了能准确和快速地排除错误, VB 提供了良好的程序调试环境和调试工具, 这些工具支持多种调试方式。

13.3.1 编辑器设置——自动语法检测

当在代码窗口输入一行有错误的语句时, VB 会自动进行语法检查, 显示出错信息。若忽略此错误, 则该行语句会以反色显示。一般在安装完 VB 后, VB 会默认设置了自动语法检测功能(也可以程序员自己设置)。

要打开编辑器窗口, 在 VB 集成开发环境中, 选择“工具”→“选项”菜单, 再在打开的“选项”对话框中选择“编辑”选项卡, 如图 13-6 所示, 在“代码设置”栏中选择“自动语法检测”项即可。

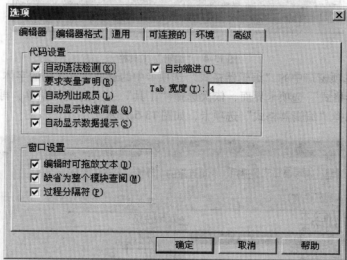


图 13-6 “选项”对话框中的“编辑器”选项卡

13.3.2 调试工具

程序的调试在整个程序实现中占有非常重要的地位, 主要任务是确定出现错误的原因, 以及出错的位置。

VB 中的调试支持包括断点、中断表达式、监视表达式、逐语句或逐过程、显示变量和属性值等。有了调试工具, 可以方便地确定程序执行的情况和各变量、表达式的值, 从而帮助编程人员更好地掌握程序的运行状态。

1. 调试工具栏

在 VB 中提供了一个调试工具栏, 以便快速地使用各种调试工具。可在 VB 工具栏中单

击鼠标右键并选择“调试”选项。“调试”工具栏如图 13-7 所示。



图 13-7 “调试”工具栏

通过“调试”工具栏提供的快捷图标按钮，可以设置程序运行、暂停及中止运行 3 种状态；可以选择设置逐语句、逐过程等运行方式；可以打开与调试相关的窗口，如立即窗口、监视窗口及本地窗口等。

2. 调试菜单

除了调试工具栏外，也可以通过 VB 中的“调试”菜单完成调试工具栏中的部分功能。

13.3.3 程序运行模式

VB 的程序有多种运行模式，不同运行模式有不同作用。VB 中主要有 4 种运行模式：全运行模式、逐语句运行模式、逐过程运行模式和断点运行模式。

1. 全程运行模式

在全程运行模式下应用程序全部编译并运行，若没有错误发生将运行出最终的结果。若执行到错误，程序将提示错误发生，若忽略，程序将继续运行。若终止，程序结束运行，在输出窗口会显示出错误的位置和原因。

VB 系统默认的运行模式为全程运行模式。这种运行模式一般用于对程序进行粗略的定位，以便进一步调试。

2. 逐语句运行模式

逐语句运行模式是在调试程序时常用的一种运行模式。使用该模式对程序代码逐语句运行，可以监测程序的每一步运行，监测各种变量、函数在程序运行中每一步的执行结果。

可以通过调试工具栏、调试菜单或快捷键 F8 进入逐语句调试。这种运行方式一般用于对程序的错误已经有了一个较小范围的定位之后，如果程序的代码很多，不建议使用该运行模式。

3. 逐过程运行模式

与“逐语句”相似，逐过程运行模式是以应用程序的过程为单位运行调试的，每次执行一个过程，而不进入过程内部执行。当调试中调用某个过程时，若过程中没有错误则跳过此过程。

通常在确认某些过程不存在错误时选用；可以通过调试工具栏、调试菜单或快捷键 Shift + F8 进入逐过程运行模式。

4. 断点运行模式

断点运行模式是一种重要的调试方式。可以在程序中加入断点，当程序运行到断点时，程序会在有断点的地方停止运行，进入中断模式。

可在代码编辑窗口中，用鼠标单击要设置断点的代码行左边框位置的方法设置断点；鼠标再次单击可清除断点。如图 13-8 所示。

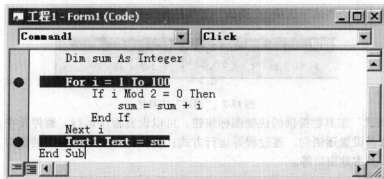


图 13-8 在程序中设置断点

13.3.4 添加监视

当程序运行发生错误时，除了通过设置断点进行调试外，还可以利用监视窗口观察程序中的变量、表达式及函数值等的变化。

监视窗口的添加方法为：依次选择菜单“调试”→“添加监视”命令，打开监视窗口对话框。

程序只有在处于中断状态下，才可以在监视窗口中观察到被监视对象的变化情况。例如，在图 13-9 所示的代码窗口中的 Command1_Click() 事件中添加对变量 sum 的监视，并设置断点。

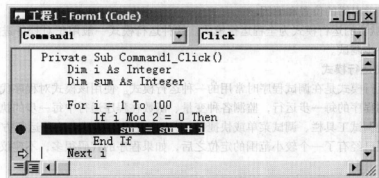


图 13-9 代码窗口

通过观察如图 13-10 所示的监视窗口，可以看到变量 sum 的变化情况。

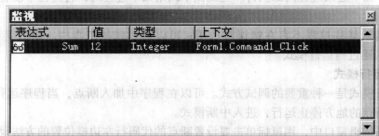


图 13-10 监视窗口

13.4 错误处理

错误是难以避免的,有时是难以发现的。为了提高程序的可靠性和稳定性,在VB中提供了处理异常错误的语句 On Error,通过在程序中加入此语句可以对程序中的错误进行处理。

VB 处理异常错误的过程如下:设置错误陷阱捕获错误;错误处理;退出错误处理程序。

1. 捕获错误

通过 On Error 语句可进行异常错误的捕获。

1) 发生错误时,转到语句标号处执行,格式如下:

On Error GoTo 语句标号

2) 发生错误时,忽略错误行,继续执行下一条语句,格式如下:

On Error Resume Next

3) 禁止当前过程中任何已启动的错误处理程序,格式如下:

On Error GoTo 0

2. 编写错误处理程序

在运行程序时,当执行到异常错误处,将转到错误处理程序,输出错误信息。

编写错误处理程序一般用到 Err 对象,它是一个系统对象。通过该对象可以获取程序的错误信息。当程序发生错误时,有关错误的信息会保存在 Err 对象中。Err 对象每次只保留当前的错误信息。Err 对象的常用的属性包括:

1) Number 属性:返回或设置表示错误的数值。Number 是 Err 对象的默认属性,该属性可读/可写。

2) Description 属性:返回或设置一个字符串表达式,包含与对象相关联的描述性字符串。此属性可读/可写。

【例 13-1】 下面的代码是编写的一个简单的除法操作程序并进行了出错的处理。

代码如下:

```
Private Sub ErrSub()  
    On Error GoTo ErrHandle  
    Dim i As Integer  
    Dim a As Integer  
    Dim div As Integer  
    a = 10  
    div = a / i  
    Exit Sub  
ErrHandle:  
    MsgBox "错误描述:" & Err.Description & ",错误号:" & Err.Number, vbCritical +  
        vbOKOnly, "错误提示"  
End Sub
```

调用此代码,则会弹出如图 13-11 所示的提示信息。

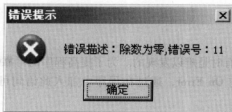


图 13-11 输出的错误提示信息

3. 退出错误处理

在错误处理程序中，当遇到 Exit Sub、Exit Function、End Sub、End Function 等语句时，将退出错误捕获。

13.5 应用程序的部署

应用程序的部署是将用户所编写的程序编译成可执行文件，然后制作成可以脱离编译环境的安装文件，以方便在其他用户的计算机上使用。在 Windows 环境下，生成的 Setup.exe 文件就是安装文件。

1. 生成工程的“.exe”文件

要生成项目的 Setup.exe 安装文件，必须首先生成工程的可执行文件，在 VB 中生成“.exe”文件，可按以下步骤进行：

1) 打开要生成“.exe”文件的工程，依次选择“文件”→“生成... .exe”，单击打开“生成工程”对话框，如图 13-12 所示。

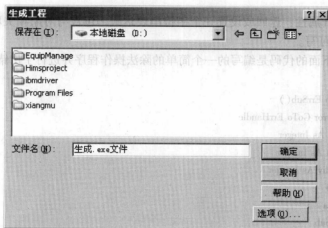


图 13-12 “生成工程”对话框

2) 若对生成的“.exe”文件的版本号、版本信息、应用程序的标题、图标等信息进行编辑，可以选择“生成工程”中的选项按钮，打开“工程属性”对话框，如图 13-13 所示。

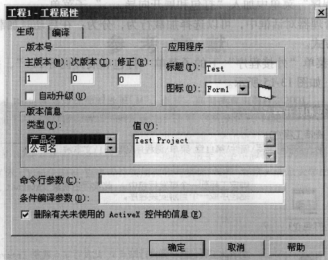


图 13-13 “工程属性”对话框

3) 单击“确定”按钮,可生成工程的“.exe”文件了。但此时的可执行文件,必须在安装了VB运行环境的计算机上才能运行。

2. 创建安装文件

若想使开发的项目被用户使用,常常需要生成项目的安装文件。VB提供了制作安装文件的工具——打包和安装向导,使用此工具就可制作安装文件了。为应用程序制作安装程序的步骤如下:

1) 为VB添加“打包和安装向导”。在VB开发环境中,依次打开菜单“外接程序”→“外接程序管理器”,单击可打开“外接程序管理器”窗口。如图13-14所示。

在“外接程序管理器”窗口中,双击“打包和展开向导”项,单击“确定”按钮,便

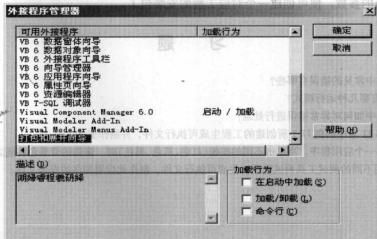


图 13-14 “外接程序管理器”窗口

可在菜单“外接程序”菜单中加入“打包和展开向导...”子菜单。

在外接程序管理器对话框中, 可以选择加载行为, 分为: 在启动时加载; 加载或卸载; 命令行形式 3 种方式。

2) 通过单击菜单“外接程序”下的子菜单“打包和展开向导...”, 打开“打包和展开向导”对话框。如图 13-15 所示。

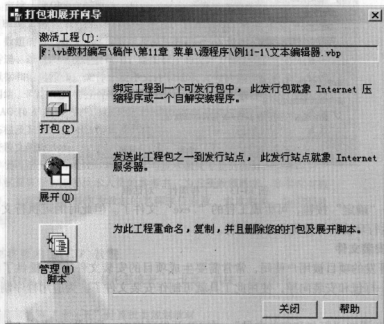


图 13-15 “打包和展开向导”对话框

3) 在“打包和展开向导”窗口, 单击“打包”按钮, 按照“打包和展开向导”的提示, 选择适当的参数, 便可创建一个对应工程的安装包了。

习 题

- 13-1 编程中常见的错误有哪些?
- 13-2 VB 有哪几种运行模式?
- 13-3 程序中如何对异常错误进行处理?
- 13-4 将第 11 章中的例 11-1 所创建的工程生成可执行文件, 并制作安装文件。
- 13-5 创建一个应用程序, 当单击不同的按钮(红色、蓝色、黑色)时, 窗体的背景色将随之改变。要求: 设计程序、使用不同的调试工具测试程序、生成可执行文件、制作此应用程序的安装文件。

参 考 文 献

- [1] 龚沛曾. Visual Basic 程序设计简明教程[M]. 北京: 高等教育出版社, 2001.
- [2] 刘炳文. Visual Basic 程序设计教程[M]. 北京: 清华大学出版社, 2002.
- [3] 徐尔贵, 丁雷. Visual Basic 教程[M]. 北京: 北京交通大学出版社, 2003.
- [4] 陈学东. Visual Basic 6.0 程序设计教程[M]. 北京: 清华大学出版社, 2005.
- [5] 陈翠松, 徐宝林. Visual Basic 程序设计实用教程与实训[M]. 北京: 北京大学出版社, 2006.